MINISTÉRIO DA SAÚDE
FUNDAÇÃO OSWALDO CRUZ
INSTITUTO OSWALDO CRUZ

Doutorado em Programa de Pós-Graduação em Biologia Computacional e Sistemas

# GENÔMICA COMPARATIVA EM AMBIENTE COMPUTACIONAL DISTRIBUÍDO: APLICABILIDADE E POTENCIAL NO ESTUDO DE HOMOLOGIA ENTRE PROTOZOÁRIOS

NELSON PEIXOTO KOTOWSKI FILHO

Rio de Janeiro
Outubro de 2015

Ministério da Saúde

FIOCRUZ
Fundação Oswaldo Cruz

**INSTITUTO OSWALDO CRUZ**
**Programa de Pós-Graduação em Biologia Computacional e Sistemas**

*NELSON PEIXOTO KOTOWSKI FILHO*

# Genômica comparativa em ambiente computacional distribuído: aplicabilidade e potencial no estudo de homologia entre protozoários

Tese apresentada ao Instituto Oswaldo Cruz como parte dos requisitos para obtenção do título de Doutor em Biologia Computacional e Sistemas

**Orientador:** Prof. Dr. Alberto Martín Rivera Dávila

**RIO DE JANEIRO**
Outubro de 2015

Ficha catalográfica elaborada pela
Biblioteca de Ciências Biomédicas/ ICICT / FIOCRUZ - RJ

K87 Kotowski Filho, Nelson Peixoto

Genômica comparativa em ambiente computacional distribuído: aplicabilidade e potencial no estudo de homologia entre protozoários / Nelson Peixoto Kotowski Filho. – Rio de Janeiro, 2015.
xvii,198 f. : il. ; 30 cm.

Tese (Doutorado) – Instituto Oswaldo Cruz, Pós-Graduação em Biologia Computacional e Sistemas, 2015.
Bibliografia: f. 67-77

1. Genômica comparativa. 2. Protozoários. 3. Ortologia. 4. Computação em nuvem. 5. Workflow científico. I. Título.

CDD 572.8633

Ministério da Saúde

**FIOCRUZ**
**Fundação Oswaldo Cruz**

## INSTITUTO OSWALDO CRUZ

**Programa de Pós-Graduação em Biologia Computacional e Sistemas**

***AUTOR:*** ***NELSON PEIXOTO KOTOWSKI FILHO***

# GENÔMICA COMPARATIVA EM AMBIENTE COMPUTACIONAL DISTRIBUÍDO: APLICABILIDADE E POTENCIAL NO ESTUDO DE HOMOLOGIA ENTRE PROTOZOÁRIOS

**ORIENTADOR: Prof. Dr. Alberto Martín Rivera Dávila**

**Aprovada em: 28/10/2015**

**EXAMINADORES:**

**Prof. Dra.** Renata Schama Lellis**– Presidente** (IOC/FIOCRUZ)
**Prof. Dr.** Fabricio Alves Barbosa da Silva (PROCC/FIOCRUZ)
**Prof. Dr.** Sérgio Manuel Serra da Cruz (UFRRJ)
**Prof. Dr.** Milton Ozório Moraes (IOC/FIOCRUZ)
**Prof. Dra.** Maria Claudia Reis Cavalcanti (IME)

Rio de Janeiro, 28 de Outubro de 2015.

Aos nossos anjos da guarda.

## AGRADECIMENTOS

À(o) grande arquiteto(a) desse universo – quiçá de outros também. Tá certo que de vez em quando a gente estranha, mas no final penso que isso é o que nos faz humanos. Você(s?) deixou uma grande lição, que é nunca deixar de sonhar. E, na boa, continua dando uma força aí, por favor...! Obrigado!!!

À minha esposa. Sempre que penso em o que é determinação, serenidade, compaixão, companhia e amor verdadeiro é para você que olho. Você entrou de cabeça na nossa aventura e só assim conseguimos chegar onde chegamos. Lembrarei disso, para todos os dias dessa e de todas as outras vidas que venham pela frente. Amo-te.

À minha família, vocês me deram a maior força e nunca desistiram de mim durante todas as ausências nestes anos. Nós sabemos o quanto gostaríamos de estar mais próximos uns dos outros, mas como por vários motivos não pudemos fazer desta maneira. Esse momento, essa passagem pela linha de chegada só acontece por conta do apoio de todos vocês. Espero que um dia eu possa dedicar essa mesma atenção e energia a quem precisar...

Ao meu orientador, Dr. Alberto Martín Rivera Dávila, por ter acreditado e investido em mim no momento em que eu mais queria uma oportunidade de estudar novamente. Fui acomodado em um laboratório nota dez, com pesquisadores e alunos que são feras!

Aos Drs. Rodrigo, Diogo e Rafael, obrigado por aliviarem minha “deficiência biológica” (Jardim, 2013(Dávila, 2010)). A paciência que vocês demonstraram ao me escutar, ensinar, repetir, repetir, repetir até eu fixar os conceitos, só vocês mesmos! Penso que devo muitas horas de paciência para vocês, Dr. Rodrigo então, vixe!

À Equipe do Laboratório de Biologia Computacional e Sistemas (LBCS). Demos muitas risadas, ouvi muitas estórias, conheci muitas pessoas bacanas, estudamos muito e fizemos bem mais que “cinco linhas de código”.

Paper, valeu pelas dicas quando eu mais precisava!

À Coordenação, Secretaria e representação discente do Programa de Pós-Graduação em Biologia Computacional e Sistemas da Fiocruz, por me apoiarem e

orientarem durante o meu processo de candidatura ao Doutorado Sanduíche e durante todo o meu curso.

À Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – CAPES pela bolsa concedida através do Programa de Doutorado Sanduíche no Exterior (PDSE).

À Fiocruz, que entendeu meu desejo de estudar e permitiu meu ingresso nesta Pós-Graduação e minha ausência durante o Doutorado Sanduíche.

Ao Dr. Ignacio Blanquer-Espert, ao docente Abel Carrión e toda a sua equipe do I3M, da Universitat Politècnica de València, por terem me recebido, me co-orientado e me aturado (sic!) no estrangeiro.

À cidade de València, que nos acolheu de tal forma que abriu nossos horizontes, fez-nos ver a vida de outra maneira e assim nos modificou de forma inesperada.

E vamos que vamos, mais uma etapa!

*“Você é chamado, despertado, empurrado, criticado, corrigido, reprimido, imprensado.*

*A vida exige sua presença, sua correção e atenção.*

*E você cambaleia, desculpa, contorna, retorna, explica, se põe de pé e segue avante.*

*Para você vencer, arrime-se no rumo e na filosofia do amor, que tudo supera.*

*Lute e seja paciente. A vitória está vindo.*

*Uma intensa luta alicerça em você a Alegria de ser firme e forte.”*

Candido Kotowski

Ministério da Saúde

**FIOCRUZ**
**Fundação Oswaldo Cruz**

# INSTITUTO OSWALDO CRUZ

## GENÔMICA COMPARATIVA EM AMBIENTE COMPUTACIONAL DISTRIBUÍDO: APLICABILIDADE E POTENCIAL NO ESTUDO DE HOMOLOGIA ENTRE PROTOZOÁRIOS

### RESUMO

**TESE DE DOUTORADO EM BIOLOGIA COMPUTACIONAL E SISTEMAS**

**Nelson Peixoto Kotowski Filho**

A inferência de homologia entre organismos é uma atividade da genômica comparativa que possibilita compreender melhor a relação entre os mesmos e, por conseguinte, sua distância evolutiva. Especificamente, a identificação de genes ortólogos, ou seja, aqueles que têm sua origem em um ancestral comum, permite oferecer melhorias na anotação funcional de genes, uma vez que genes ortólogos tendem a ter sua função conservada.

Com a crescente disponibilidade de genomas através de técnicas de NGS, a construção e atualização de bases de dados de ortólogos representam um desafio constante, pois demandam o estudo e identificação das relações entre os genes de tais organismos, em um volume de dados cada vez mais extenso e a um custo computacional cada vez mais elevado.

Nesta tese propomos a solução para nuvem computacional *elastic-OrthoSearch*, um *workflow* científico de genômica comparativa inspirado no *OrthoSearch*, responsável pela inferência de homologia entre organismos com o uso de abordagem baseada em melhores *hits* recíprocos e perfis de *Markov*.

Também propomos uma metodologia para criação de bases de ortólogos construída através do reuso do *OrthoSearch*. Esta metodologia mostrou-se capaz de alavancar a oferta de grupos ortólogos e assim auxiliar, por exemplo, na identificação de alvos de protozoários.

Ministério da Saúde

FIOCRUZ
Fundação Oswaldo Cruz

# INSTITUTO OSWALDO CRUZ

## COMPARATIVE GENOMICS IN A DISTRIBUTED COMPUTING SCENARIO: APPLICABILITY AND POTENTIAL ON PROTOZOA HOMOLOGY STUDY

### ABSTRACT

**PHD THESIS IN COMPUTATIONAL SYSTEMS AND BIOLOGY**

**Nelson Peixoto Kotowski Filho**

Homology inference among organisms is a comparative genomics tasks which allows for a better understanding on how such organisms are related to each other and on their evolutionary distance. Specifically, the identification of orthologous genes – those who share a common ancestor – allows for functional gene annotation improvements, as orthologous genes tend to preserve their functions.

The increasing amount of genomic data provided by the NGS techniques makes the orthologous databases' building and update processes a challenging task. It requires the identification and study of the organisms' genes relationships, in an extensive data volume and at an increasing computational cost.

In this thesis we propose elastic-OrthoSearch, a cloud-enabled comparative genomics scientific workflow, derived from OrthoSearch. It aims at providing homology inference among organisms, in a reciprocal best hits and Markov profiles approach.

We also propose an improved orthologous database creation methodology built on top of OrthoSearch. Such methodology has shown means to offer a broader orthologous groups dataset, which could in turn aid on Protozoa target identification.

# ÍNDICE

# ÍNDICE DE FIGURAS

# LISTA DE TABELAS

## LISTA DE SIGLAS E ABREVIATURAS

| | |
|---|---|
| CAPES | Coordenação de Aperfeiçoamento de Pessoal de Nível Superior |
| CSV | do inglês *Comma-Separated Values* |
| DNDi | do inglês *Drugs for Neglected Diseases Initiative* |
| DAG | do inglês *Direct Acyclic Graph* |
| DST | Doença Sexualmente Transmissível |
| DTN | Doença Tropical Negligenciada |
| EC2 | do inglês *Elastic Computing Cloud* |
| GUS | do inglês *Genomics Unified Schema* |
| I3M | do espanhol *Instituto de Instrumentación para Imagen Molecular* |
| IaaS | do inglês *Infrastructure as a Service* |
| IOC | Instituto Oswaldo Cruz |
| JSON | do inglês *JavaScript Object Notation* |
| KO | do inglês *Kegg Orthology* |
| LBCS | Laboratório de Biologia Computacional e Sistemas |
| LC | Leishmaniose cutânea |
| MCL | do inglês *Markov Cluster Algorithm* |
| Mpb | Mega pares de bases |
| NFS | do inglês *Network File System* |
| NGS | do inglês *Next-Generation Sequencing* |
| OMS | Organização Mundial de Saúde |
| PaaS | do inglês *Platform as a Service* |
| PDSE | Programa de Doutorado Sanduíche no Exterior |
| RSD | do inglês *Reciprocal Smallest Distance* |
| SaaS | do inglês *Software as a Service* |
| SPIMAP | do inglês *Species Informed Maximum A Posteriori Gene Tree Reconstruction* |
| SWfMS | do inglês *Scientific Workflow Management System* |
| UPV | do valenciano *Universitat Politècnica de València* |
| WfMC | do inglês *Workflow Management Coalition* |

# 1 INTRODUÇÃO

## 1.1 Protozoários

### *1.1.1 Sobre Protozoários*

A classificação de organismos como protozoários é, historicamente, tema de debate desde as primeiras referências a tal termo (Cavalier-Smith, 2009a; Cavalier-Smith, 2009b; Cavalier-Smith, 1993; Imam, 2009).

Enquanto Kotpal (Kotpal, 2012) os define como um sub-reino do reino *Animalia*, composto de organismos eucariotos unicelulares, Cavalier-Smith (Cavalier-Smith, 2009a; Cavalier-Smith, 2009b; Cavalier-Smith, 1993) detalha essa classificação, estendendo-a aos organismos eucariotos que têm mitocôndria, peroxissomas e demais características compartilhadas que os separam dos reinos *Animalia*, *Fungi*, *Plantae* e *Chromista*.

De acordo com Imam (Imam, 2009), existem mais de 200.000 organismos nomeados como protozoários, dos quais praticamente todos parasitam vertebrados e 10.000 parasitam invertebrados.

### *1.1.2 Doenças Tropicais Negligenciadas*

A Organização Mundial de Saúde (OMS) mantém diversos programas para acompanhamento da pesquisa e desenvolvimento de medicamentos para as doenças que afetam a população mundial. A Comissão em Saúde e Macroeconomia da OMS, apoiada em documento previamente publicado pela organização humanitária Médicos Sem Fronteiras (MSF, 2001), classifica as doenças em: Tipo I, Tipo II ou Tipo III (de Souza et al., 2010; WHO, 2010a).

As doenças de Tipo I, presentes em países desenvolvidos e não desenvolvidos, são usualmente endereçadas pelas grandes companhias fabricantes de medicamentos e vacinas e também são alvo de pesquisa e desenvolvimento constante.

Por outro lado, as doenças dos tipos II e III recebem menor investimento do que o projetado pela OMS, o que pode ser associado ao fato de serem majoritariamente endêmicas em regiões e países pobres ou em desenvolvimento, o que teoricamente representa um problema de mercado para as grandes empresas.

As denominadas Doenças Tropicais Negligenciadas (DTNs) encaixam-se nos tipos II e III. São ocasionadas por uma variedade de organismos e usualmente associadas a padrões deficitários de higiene, saneamento, sociais e financeiras e por isso, presentes em sua maioria em países e regiões do globo em desenvolvimento.

As DTNs afetam mais de hum bilhão de pessoas em 149 países ao redor do mundo e dentre as 17 doenças classificadas como DTNs pela OMS, listadas na

Tabela 1.1, três têm protozoários como patógenos: Doença de Chagas (*Trypanosoma cruzi*), Tripanossomíase Africana (*Trypanosoma brucei*) e Leishmaniose (*Leishmania* spp.) (WHO, 2012a).

Tabela 1.1: Doenças Tropicais Negligenciadas e natureza de seus patógenos, segundo classificação da OMS. (Fonte: adaptado de (WHO, 2012a)).

| **Enfermidade** | **Natureza do Patógeno** |
|---|---|
| Doença de Chagas | Protozoário |
| Tripanossomíase Humana Africana | Protozoário |
| Leishmaniose | Protozoário |
| Úlcera de Buruli | Bactéria |
| Hanseníase | Bactéria |
| Tracoma | Bactéria |
| Bouba | Bactéria |
| Neurocisticercose/Teníase | Helmintos |
| Dracucunlíase | Helmintos |
| Equinococose | Helmintos |
| Infecções por tremátodes | Helmintos |
| Filariose linfática | Helmintos |
| Oncocercose | Helmintos |
| Esquistossomose | Helmintos |
| Helmintíases transmitidas pelo solo (Ascaridiose, Tricuríase, Estrongiloidíase) | Helmintos |
| Dengue e Chikungunya | Vírus |
| Raiva | Vírus |

De acordo com o terceiro relatório da OMS sobre as DTNs, apesar dos diversos avanços nos anos recentes, a atenção para com a pesquisa e inovação no diagnóstico, tratamento e prevenção para tais doenças deve ser permanente (Uniting to Combat NTDs, 2014; WHO, 2015a).

A pesquisa e desenvolvimento de medicamentos para o tratamento das DTNs é o foco da organização DNDi (DNDi, 2015), que compreende sete instituições, sendo cinco do setor público (Fundação Oswaldo Cruz, Conselho de Pesquisa Médica da Índia, Instituto de Pesquisa Médica do Quênia, Ministério da Saúde da Malásia e o Instituto Pasteur), uma organização humanitária (Médicos Sem Fronteiras) e uma organização de pesquisa internacional (o Programa Especial para Pesquisa e Treinamento em Doenças Tropicais – TDR) da OMS.

### *1.1.3 Protozoários e a saúde humana e de outros organismos*

Protozoários são organismos de importante relevância para a saúde humana e de demais espécies. Além das três enfermidades causadas por protozoários listadas no terceiro Relatório sobre as Doenças Tropicais Negligenciadas (WHO, 2015a) - Doença de Chagas, Tripanossomíase Africana e Leishmaniose – o Programa Especial para Pesquisa e Treinamento em Doenças Tropicais (TDR) da OMS também lista a malária, ocasionada por protozoários do gênero *Plasmodium*, como prioridade de investigação (WHO, 2014).

Outras enfermidades também associadas a protozoários apresentam impacto relevante não somente na saúde humana, mas em outras espécies, como no caso da giardíase, toxoplasmose, babesiose, theileriose, criptosporidíase, amebíase e tricomoníase (Brayton et al., 2007; Carlton et al., 2007; Elmore et al., 2010; Lorenzi et al., 2010; Pain et al., 2005; Thompson, 2009; Widmer et al., 2009).

Juntas, estas doenças representam elevado perigo para a população mundial o que as torna candidatas atrativas para pesquisas que possam aprimorar o conhecimento das espécies que as ocasionam e suas vias metabólicas pertinentes, de forma que seja possível contribuir para o tratamento das mesmas.

A malária é uma doença resultante de uma infecção por parasitas do gênero *Plasmodium*. É transmitida por mosquitos infectados do gênero *Anopheles* (Haldar & Mohandas, 2009) e é a doença parasitária mais prevalente no mundo, com mais de 216 milhões de casos reportados. O padrão global de transmissão da malária sugere

uma doença centrada nos trópicos, mas com alcance em regiões subtropicais nos cinco continentes (Sachs & Malaney, 2002).

Dentre os plasmódios que infectam predominantemente os seres humanos, o *P. falciparum* é o maior responsável pelos casos de malária (Wellems et al., 2009), além das espécies *P. vivax*, *P. ovale* e *P. malariae* (Ollomo et al., 2009). Recentemente, casos associados ao parasita *P. knowlesi* atraem atenção pelos crescentes números observados (Hall & Carlton, 2005).

A malária que ocorre em roedores, através das espécies *P. chabaudi*, *P. yoelii*, e *P. Berghei* (Carlton et al., 2002; Pain et al., 2008) é utilizada como modelo e campo experimental de pesquisa para desenvolvimento de terapias e melhor compreensão do *P. falciparum*, uma vez que os parasitas da malária humana são de difícil cultivo em laboratório (Killick-Kendrick et al., 1978).

A toxoplasmose é uma enfermidade provocada pelo protozoário *Toxoplasma gondii*; estima-se que infecte aproximadamente um terço da espécie humana (Weiss & Dubey, 2009) e também infecta outros animais homeotérmicos, sendo o gato (*Felix catus*) considerado o hospedeiro definitivo.

Com diversos casos emergentes na espécie humana, a babesiose, provocada pelo protozoário *Babesia bovis*, usualmente afeta outros mamíferos (Brayton et al., 2007; Hunfeld et al., 2008) e é transmitida através do carrapato *Ixodes scapularis*. Seus impactos na saúde incluem febre e anemia severa.

Os protozoários *Theileria annulata* e *Theileria parva* são responsáveis por ocasionar duas enfermidades associadas ao gado: a Febre da Costa do Leste e a Theileriose Tropical. São hemoparasitas e causam febre e anemia, entre outros sintomas. Seus vetores são carrapatos do gênero *Hyalomma*. As medidas terapêuticas atuais incluem a administração de vacinas com parasitas vivos e são confiáveis apenas nas fases iniciais da doença (Pain et al., 2005).

Os protozoários do gênero *Cryptosporidium* completam seu ciclo de vida em um único hospedeiro, sendo transmitidos por água ou comida contaminada; ocasionam diarreia e gastroenterite aguda. Afetam diversas espécies de mamíferos, além da humana, na qual a maioria dos casos está relacionada à *C. hominis* e *C. parvum*, mas outras linhagens como *C. muris* também são capazes de provocar a enfermidade (Abrahamsen et al., 2004; Widmer et al., 2009; Xu et al., 2004).

O *Trypanosoma cruzi* é o agente etiológico da Doença de Chagas, grave problema de saúde e principal causa de cardiopatia na América Latina, atinge mais de 10 milhões de pessoas, com mais de 10 mil óbitos até 2008 (Lewis et al., 2009;

WHO, 2010b). Recentemente vem sendo detectada nos Estados Unidos, Canadá, em alguns países europeus e do Oeste do Pacífico (Vallejo et al., 2002; WHO, 2010b).

O *Trypanosoma brucei*, parasita transmitido através da picada da mosca hematófaga Tsé-tsé (*Glossina* spp.), é o agente etiológico da "enfermidade do sono". Estimativas recentes apontam cerca de 30 mil casos desta doença, que expõe principalmente habitantes da África subsaariana, em especial as populações rurais, as expostas à extrema pobreza e/ou guerra (Barrett et al., 2003; WHO, 2012b).

O *Trypanosoma vivax* pode ser transmitido aos animais domésticos pela mosca Tsé-tsé (*Glossina* spp.) e já se espalhou para dez dos treze países da América do Sul (Jones & Dávila, 2001). Representa um risco potencial para aproximadamente 300 milhões de bovinos, 1,8 milhões de búfalos e 16 milhões de cavalos. A transmissão dessa doença é debilitante para a criação de gado de corte e de tração (Dávila et al., 2003).

A leishmaniose é uma doença zoonótica causada por protozoários do gênero *Leishmania*, transmitida para seus hospedeiros através da picada de flebotomíneos fêmea. Encontra-se amplamente distribuída em todo o mundo e já afeta aproximadamente 12 milhões de pessoas (El-On, 2009; WHO, 2015b), com estimativas de dois milhões de novos casos e de 20.000 a 30.000 mortes anuais (WHO, 2015b; WHO, 2015c). Está usualmente relacionada à extrema pobreza, é mais prevalente em países em desenvolvimento e causa grande espectro de doenças nos seres humanos (CDC, 2015; El-On, 2009; WHO, 2015c).

A leishmaniose cutânea (LC) é causada por *Leishmania major*, *Leishmania tropica* e *Leishmania aethiopica* no Velho Mundo, e *Leishmania mexicana*, *Leishmania braziliensis* no Novo Mundo. Caracteriza-se pelo desenvolvimento de lesões ulcerosas na pele (Grimaldi & Tesh, 1993). A forma mucocutânea é uma complicação da LC, com parasitas disseminando-se através do sistema linfático para colonizar o trato da mucosa. Os pacientes sofrem de úlceras destrutivas da mucosa, que os desfiguram de forma permanente.

O tipo cutâneo difuso manifesta-se quando a LC se dissemina em pacientes com imunidade defeituosa das células do tipo T. Responde mal aos medicamentos e provoca feridas ou úlceras por todo corpo. A forma visceral é caracterizada por uma infecção sistêmica do fígado e baço, causada pelas espécies: *Leishmania donovani* e *Leishmania infantum* (Grimaldi & Tesh, 1993; Murray et al., 2005).

Existem outras espécies de protozoários que também apresentam impacto relevante na saúde humana e de outros mamíferos e que possuem distribuição cosmopolita. Estes incluem, por exemplo:

- *Entamoeba histolytica* e *Entamoeba dispar*: associados à amebíase, que provoca diarreia e disenteria (Lorenzi et al., 2010), parasita o intestino humano;
- *Giardia lamblia*: responsável pela transmissão da giardíase, associado a hábitos de higiene deficientes. Infecta humanos e outros mamíferos (Adam, 2000). Provocam diarreia, náuseas, vômitos, entre outros sintomas (Morrison et al., 2007; Thompson, 2009);
- *Trichomonas vaginalis*: associado à tricomoníase, doença sexualmente transmissível (DST) sem concentração geográfica definida, apresenta em torno de 170 milhões de casos anuais (Carlton et al., 2007).

## 1.2 **Genômica Comparativa e Inferência de Homologia**

### *1.2.1 Homologia*

Com a disponibilidade crescente de genomas completos de diversos organismos, estudos funcionais sobre os mesmos permitem a observação de características relacionadas à saúde, traços e fenótipos de cada uma destas, assim como analisar a dinâmica evolutiva entre espécies distintas, frente à conservação de genes entre as mesmas, por exemplo.

Nesse contexto identificamos o conceito de genômica comparativa, que se baseia principalmente no estudo da homologia e da dinâmica evolutiva dos organismos, seus respectivos genes e proteínas, o que apresenta grande utilidade no entendimento da evolução das espécies pela comparação de seus genomas completos ou de genes específicos de cada espécie (Hardison, 2003).

Proteínas homólogas compartilham uma origem em comum, seja esta inter ou intra espécies. Existem diversos cenários relacionados à homologia, tais como: ortologia, paralogia, transferência horizontal de genes, perda gênica, genes órfãos, entre outros. Nesta tese, temos interesse especial no estudo de ortólogos.

Pode-se inferir a ocorrência de ortologia quando genes ou proteínas estão presentes em duas (ou mais) espécies e decorrem de evento de especiação. Parálogos são gerados por duplicação gênica e embora usualmente pertencentes a uma mesma espécie, podem também ocorrer entre espécies diferentes (Koonin,

2005). A Figura 1.1 e a Figura 1.2 demonstram exemplos das relações de homologia, com destaque para as definições de ortólogos e parálogos.

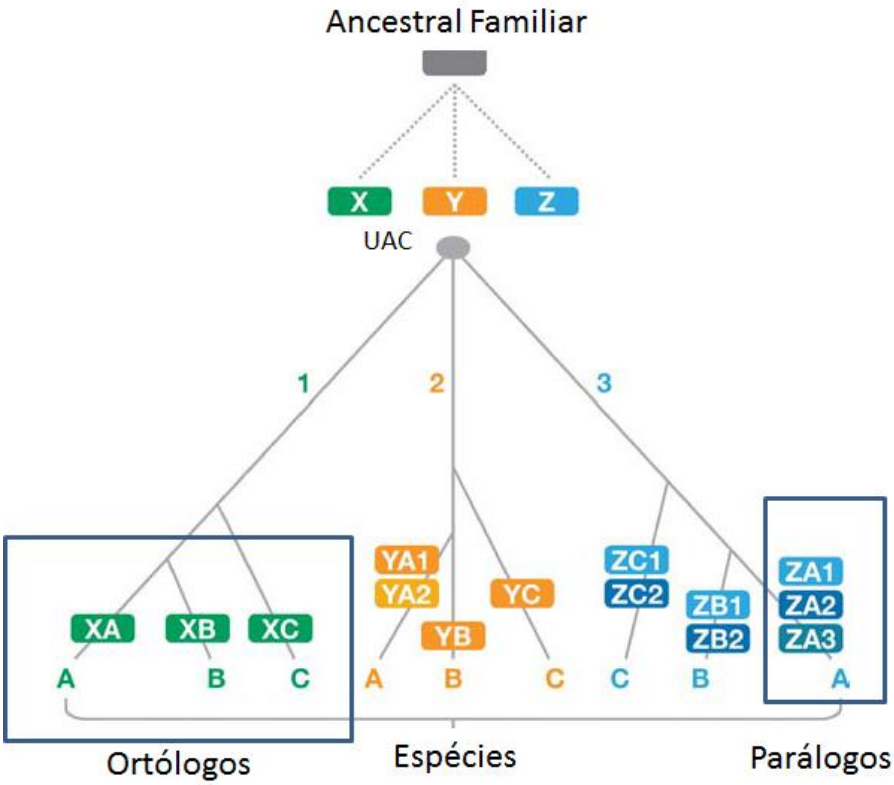


Figura 1.1: Relação hipotética de homologia, considerando as espécies A, B e C, os genes ancestrais X, Y e Z e suas variações ortólogas (especiação) e parálogas (duplicação gênica). (Fonte: adaptado de (Koonin, 2005))

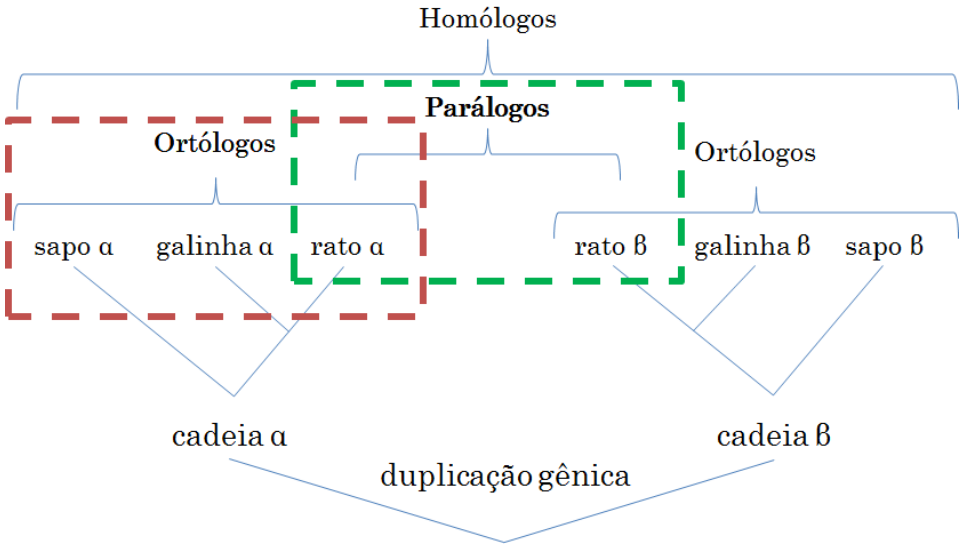


Figura 1.2: Relação de homologia referente aos genes das cadeias α e β globina, com destaque para a definição de ortólogos (especiação) e parálogos (duplicação gênica) (Fonte: adaptado de http://www.ncbi.nlm.nih.gov/Education/BLASTinfo/Orthology.html).

### *1.2.2 Inferência de homologia*

A identificação de homologia tornou-se importante para a inferência de função de genes recém-sequenciados a partir de genes experimentalmente caracterizados pelo fato de ortólogos conservarem a função ancestral, além de trazer um olhar sobre a história evolutiva dos genes e consequentemente dos organismos a estes associados (Koonin & Galperin, 2003; Koonin, 2005).

Existem diversos métodos disponíveis para auxiliar na detecção de homólogos; à parte de uma tentativa de categorização dos mesmos (Kristensen et al., 2011), seguiremos a proposta de Dalquen e colaboradores (Dalquen et al., 2013), ao enxergar três macro abordagens: os que utilizam scores de alinhamentos múltiplos com melhores *hits* recíprocos, como no OrthoSearch (da Cruz et al., 2010), OrthoMCL (Li et al., 2003), InParanoid (Remm et al., 2001), entre outros; os que trabalham com ferramentas de cálculo de distância evolutiva, tal qual o RSD (Wall et al., 2003; Wall & Deluca, 2007); e aqueles que utilizam-se da reconstrução de árvores filogenéticas, como o SPIMAP (Rasmussen & Kellis, 2011).

Esta ampla oferta de possibilidades demonstra que existe o interesse na identificação de homologia, como estas podem ser distintas e como há campo para se explorar novas abordagens ou refinar técnicas atuais.

Não obstante o método utilizado para a inferência de homologia há de se ressaltar a importância (dependência) das respectivas bases de dados utilizadas como parâmetro de entrada, para a identificação posterior de ortólogos e/ou parálogos. Este fato atrai atenção para aspectos como qualidade, quantidade e variabilidade dos dados disponíveis para a realização de experimentos.

Um dos benefícios associados ao agrupamento de proteínas em grupos ortólogos está na inferência de função, em especial quando são utilizados dados recém-sequenciados (Dessimoz et al., 2012). Além disso, os grupos ortólogos podem auxiliar na análise funcional e evolutiva dos organismos (Delsuc et al., 2005; Li et al., 2003).

Adicionalmente, a inferência de homologia tem papel importante na anotação genômica, identificação de famílias de proteínas, reconstrução de árvores filogenéticas, entre outras atividades (Cuadrat et al., 2014; Dessimoz et al., 2012; Dessimoz, 2011; Kristensen et al., 2011; Li et al., 2003).

Por fim, citamos que com a crescente disponibilidade de dados de homologia em bases públicas, oriundos de novas metodologias de sequenciamento de DNA/RNA, torna-se possível também a realização de experimentos e estudos que tenham por objetivo a identificação de novos alvos de drogas para auxiliar na produção e/ou reuso de medicamentos (Brotherton et al., 2013; Jardim, 2013; Singh et al., 2013; Tschoeke et al., 2014).

## 1.3 **Bancos de Dados para Inferência de Homologia**

Na literatura, é possível identificar a existência de diversas metodologias que deram origem a bancos de dados para inferência de homologia. Dentre os mesmos podemos citar o Kegg Orthology (KO) (Kanehisa et al., 2012; Kegg Orthology, 2012), EggNOG (Jensen et al., 2008; Powell et al., 2013), OrthoMCLDB (Chen et al., 2006), ProtozoaDB (Dávila et al., 2008), COG/KOG (Tatusov et al., 2003), Roundup (DeLuca et al., 2012), entre outros.

Cada uma destas tem sua origem em algoritmos e/ou *pipelines* distintos para inferência de homologia, que foram utilizados no agrupamento e construção de grupos ortólogos.

### *1.3.1 Kegg Orthology (KO)*

A base de dados KO é parte componente do sistema KEGG – Kyoto Encyclopedia for Genes and Genomes (Kanehisa et al., 2012; Kegg Orthology, 2012). Seus grupos ortólogos têm representatividade em toda a árvore da vida e foram criados de forma manual, por similaridade de sequências e pela análise funcional dos seus membros, que devem pertencer a uma via metabólica conservada ou por um complexo molecular comum.

Os agrupamentos de ortólogos são inferidos por um algoritmo que compara grafos de forma heurística, genoma contra via metabólica e genoma contra genoma.

### *1.3.2 EggNOG*

O banco de dados de ortólogos EggNOG (Jensen et al., 2008; Powell et al., 2013), em sua versão 4.0, é composto por grupos de ortólogos inferidos a partir de 3.686 organismos, oriundos dos três domínios da árvore da vida. No total, aproximadamente 9,6 milhões de proteínas foram agrupadas em 1,7 milhão de grupos ortólogos.

Os genomas obtidos através de fontes como o Ensembl (Flicek et al., 2012), Uniprot (The UniProt Consortium, 2013), GiardiaDB (Aurrecoechea et al., 2009), entre outras fontes, são alinhados com base no algoritmo FASTA (Pearson, 1990).

As anotações para cada proteína foram realizadas de forma automática, através de heurística que recupera as descrições existentes em texto livre de outras bases existentes (KO, COG, KOG, entre outras) e quando não identifica uma anotação, realiza a busca através do Gene Ontology (Ashburner et al., 2000).

Os grupos de ortólogos do EggNOG também contam com parálogos internos, identificados através de modelos de *Markov* construídos de acordo com marcadores genéticos curados universais previamente descritos (Ciccarelli, 2006; Sorek et al., 2007).

Para estes modelos, aplica-se um ponto de corte (de 70 a 99%) de similaridade, que definirá a alocação ou não de uma proteína como paráloga em um grupo. Os grupos são por fim construídos com base na identificação de melhores *hits* recíprocos, através da triangulação de resultados entre espécies, três a três. A Figura 1.3 ilustra os níveis taxonômicos e respectivos agrupamentos ortólogos criados pelo EggNOG 4.0, a versão atual disponibilizada como estável.

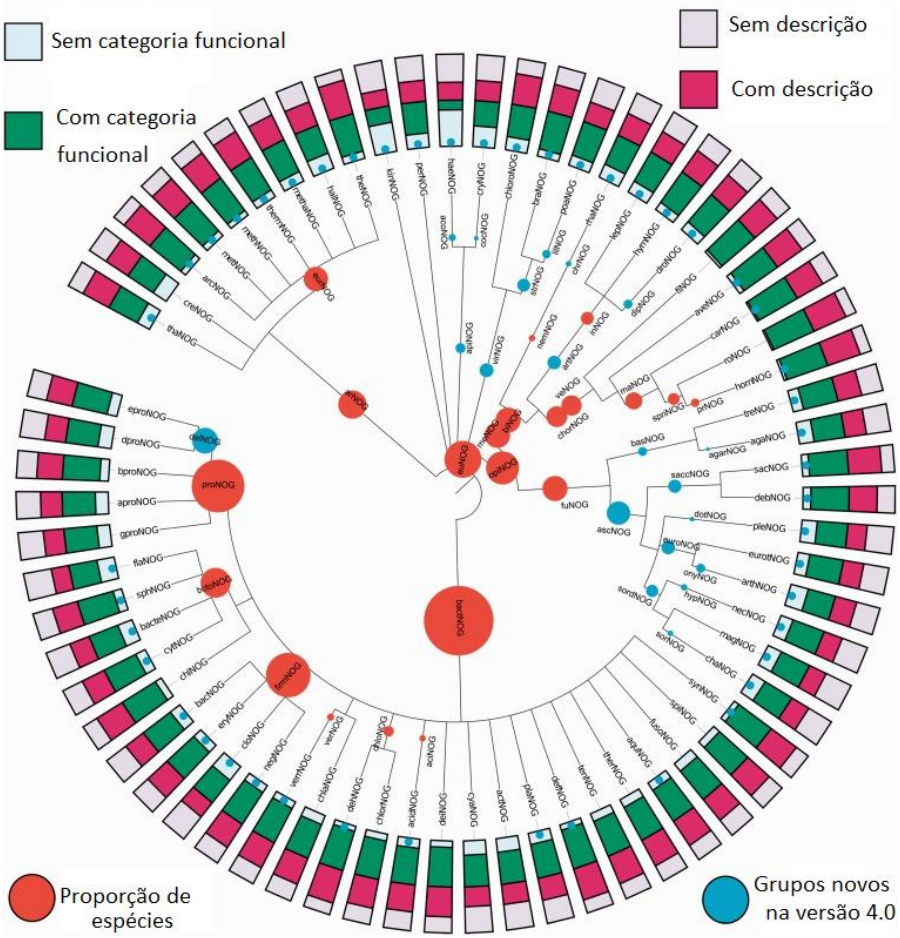


Figura 1.3: Níveis taxonômicos propostos para os grupos ortólogos do EggNOG em sua versão 4.0. (Fonte: adaptado de (Powell et al., 2013)).

### *1.3.3 OrthoMCLDB*

A base OrthoMCLDB (Chen et al., 2006) foi construída a partir de resultados de inferência de homologia do algoritmo OrthoMCL. Seus grupos contam com proteínas ortólogas e parálogas, identificadas e agrupadas através de abordagem de melhores hits recíprocos calculados por BLAST e pelo algoritmo de cadeias de Markov (Enright, 2002).

As proteínas são agrupadas com base na similaridade de sequências, calculada por um BLAST todos contra todos para cada par de genomas. As proteínas identificadas como relacionadas são interligadas em um grafo de similaridade, onde o peso entre arestas é o valor do BLAST para similaridade. Este grafo é submetido ao MCL para a criação de grupos ortólogos com base no peso entre as arestas, com a alocação de parálogos recentes nos referidos grupos, conforme apresentado na Figura 1.4.

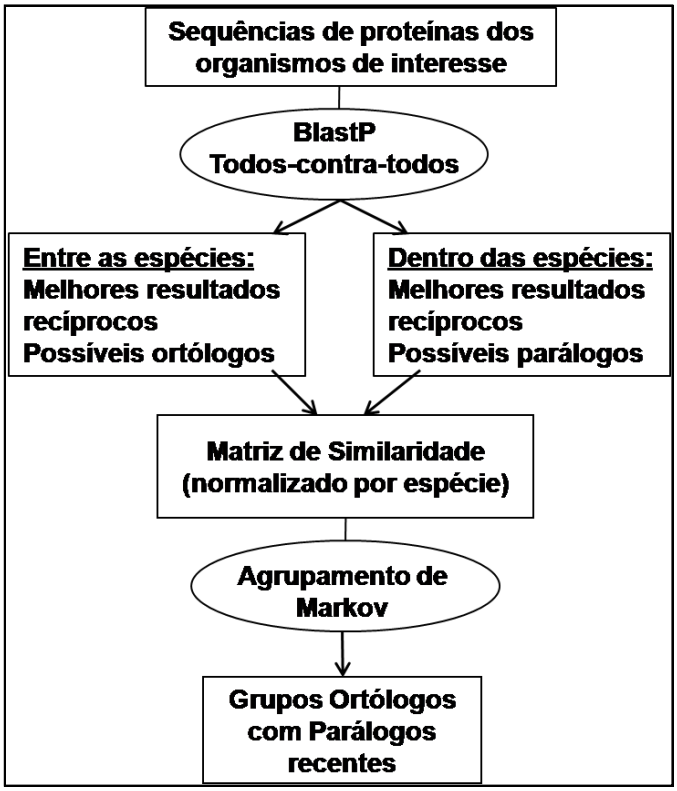


Figura 1.4: Fluxograma do algoritmo OrthoMCL, criado para o agrupamento de proteínas ortólogas, que serão depois armazenadas na base OrthoMCLDB. (Fonte: adaptado de (Li et al., 2003; Tschoeke, 2013).

O OrthoMCLDB conta atualmente com mais de 124.000 grupos de ortólogos, inferidos a partir do genoma de 150 organismos.

### *1.3.4 ProtozoaDB*

ProtozoaDB (Dávila et al., 2008) é uma suíte de ferramentas, acessível pela *internet*, para consulta a dados genômicos de protozoários, tais como: domínios conservados, vias metabólicas, identificação de ortólogos, entre outras ações.

Utiliza o *Genomics Unified Schema* (GUS) (GUS, 2012) para carga e armazenamento das sequências de nucleotídeos de tais organismos. Além disso, contemplou a construção de uma base de grupos ortólogos para os mesmos, através do algoritmo OrthoMCL (Li et al., 2003).

Em sua versão inicial, contemplou cinco organismos: *Plasmodium falciparum*, *Entamoeba histolytica*, *Trypanosoma brucei*, *Trypanosoma cruzi* e *Leishmania major*.

Atualmente, esta base conta com total de 22 organismos (todos protozoários) conforme

Tabela 1.2.

Tabela 1.2: Organismos presentes na base ProtozoaDB, em sua versão mais recente.

| **Organismo** | **Cepa** | **Identificador de Taxonomia no NCBI** |
|---|---|---|
| *Babesia bovis* | T2Bo | 484906 |
| *Cryptosporidium hominis* | TU502 | 353151 |
| *Cryptosporidium muris* | RN66 | 441375 |
| *Entamoeba dispar* | SAW760 | 370354 |
| *Entamoeba histolytica* | HM-1:IMSS | 294381 |
| *Giardia lamblia* | ATCC 50803 | 184922 |
| *Leishmania amazonensis* | MHOM/BR/1973/M2269 | 5659 |
| *Leishmania braziliensis* | MHOM/BR/75/M2904 | 420245 |
| *Leishmania infantum* | JPCM5 | 435258 |
| *Leishmania major* | Friedlin | 347515 |
| *Plasmodium berghei* | ANKA | 5823 |
| *Plasmodium chabaudi* | chabaudi | 31271 |
| *Plasmodium falciparum* | 3D7 | 36329 |
| *Plasmodium knowlesi* | H | 5851 |
| *Plasmodium vivax* | Sal-1 | 126793 |
| *Plasmodium yoelii* | yoelli 17NXL | 352914 |
| *Theileria annulata* | Ankara | 353154 |
| *Theileria parva* | Muguga | 333668 |
| *Toxoplasma gondii* | ME49 | 508771 |
| *Trichomonas vaginalis* | G3 | 412133 |
| *Trypanosoma brucei* | TREU927 | 185431 |
| *Trypanosoma cruzi* | CL-Brener | 353153 |

## 1.4 ***Workflows* Científicos**

Os primeiros esforços para estabeler a definição do conceito de *workflow* remetem à década de 1970 e à necessidade de organizações automatizarem a execução de conjuntos de tarefas através do uso de computação (Aalst & Hee, 2002).

A organização sem fins lucrativos WfMC (WFMC, 2015), referência para a padronização de termos relacionados a *workflows*, define o conceito de *workflow* como “A automação de um processo de negócio, parcial ou totalmente, durante o qual documentos, informações ou tarefas são transmitidas de um participante à outro para que determinada(s) ação(ões) seja(m) tomada(s), de acordo com um conjunto de regras” (WFMC, 1999).

Outra definição objetiva e resumida para o conceito de *workflow* é a representação lógica de uma sequência de tarefas encadeadas para atingir um objetivo final, criado de forma a garantir sua reprodutibilidade futura. (Joosten & Brinkkemper, 1995; Miller et al., 1995).

Um *workflow* científico, independente da área da ciência ao qual o mesmo se relaciona (biologia, astronomia, física, entre outras) é usualmente denominado como tal por lidar com volumes de dados extensos e heterogêneos e por requerer extensa capacidade computacional (Oliveira, 2012; Taylor et al., 2007).

O mapeamento de experimentos científicos na forma de *workflows* científicos tem sido extensivamente explorado, sendo estes usualmente suportados através da adoção de Sistemas de Gestão de *Worfklows* Científicos (SWfMS) (Mattoso et al., 2010; Taylor et al., 2007). Na literatura, podemos identificar SWfMS como o Galaxy (Goecks et al., 2010), Taverna (Oinn et al., 2004), Kepler (Altintas et al., 2004; Ludäscher et al., 2006), Vistrails (Bavoil et al., 2005), entre outros.

Idealmente, os SWfMS buscam não somente viabilizar a execução de rotinas associadas a experimentos científicos, mas também agregar características como o suporte a paralelismo e distribuição de tarefas, proveniência de dados utilizados, utilização de infraestrutura computacional heterogênea, o monitoramento das atividades, a facilidade na edição de tarefas e a reprodutibilidade dos experimentos, entre outros aspectos (Freire et al., 2008; Mattoso et al., 2010; de Oliveira et al., 2010).

O uso destes SWfMS em ambientes computacionais distribuídos pode auxiliar na melhoria de desempenho e consequentemente, prover resultados em menor intervalo de tempo do que quando o mesmo fluxo é executado de maneira serializada (de Oliveira et al., 2010; Ruberg et al., 2007) .

Com relação ao ciclo de vida de um *workflow* científico, para esta tese, seguimos a abordagem de Taylor e colaboradores (Taylor et al., 2007) na qual um protocolo ou experimento científico (por exemplo, um conjunto de *scripts* ou uma sequência de chamadas a aplicativos de bioinformática) é inicialmente projetado na

forma de um *workflow* abstrato, independente de SWfMS. Em seguida, é traduzido para atender aos requisitos de plataforma computacional (*hardware* e *software*) de acordo com o SWfMS utilizado, tornando-se assim um *workflow* concreto.

## 1.5 **Computação em Nuvem**

A nuvem computacional representa um paradigma específico de computação distribuída, no qual recursos são usualmente virtualizados e provisionados de forma dinâmica e sob demanda. Segundo Mell e Grance (Mell & Grance, 2011; NIST, 2015), tal ambiente compreende cinco características essenciais, três modelos de serviço e quatro possibilidades de provisionamento. As cinco características compreendem:

- Alocação de recursos sob demanda: refere-se à capacidade do usuário da nuvem solicitar diretamente (sem a necessidade de interação humana junto a seu provedor) recursos computacionais, conforme sua necessidade.
- Acessibilidade via rede: o usuário deve ser capaz de acessar os recursos da nuvem através de dispositivos variados (estações de trabalho, *tablets*, *laptops*, entre outros) e através da rede.
- Conjunto de recursos: os ativos computacionais que compõem a nuvem não estão necessariamente alocados em um mesmo sítio e quando disponibilizados aos usuários, podem ser solicitados em recursos físicos ou virtuais distintos, de tal forma que ao usuário não compete a ciência de onde os mesmos provêm.
- Elasticidade: a nuvem deve ser capaz de prover mecanismos que permitam manipular os recursos computacionais alocados para cada aplicação em execução. Deve ser possível entregar maior ou menor capacidade computacional quando necessário, solicitar nós adicionais e liberar nós indesejados.
- Monitoramento dos serviços: como os recursos são finitos e a demanda pelos mesmos não é conhecida *a priori*, é necessário manter uma rotina de monitoramento recursos como processamento, armazenamento, largura de banda, entre outros.

Os serviços oferecidos através de uma nuvem computacional (Höfer & Karagiannis, 2011; Mell & Grance, 2011) podem ser ofertados na forma de:

- Infraestrutura como Serviço (*Infrastructure as a Service – IaaS*): esta configuração fornece ao usuário a possibilidade de gerir recursos como os sistemas operacionais, armazenamento de dados e as aplicações disponibilizadas, geralmente sob a forma de máquinas virtuais.
- Plataforma como Serviço (*Platform as a Service – PaaS*): neste caso, o consumidor de recursos não gere a plataforma computacional subjacente, mas tem controle sob as aplicações em execução na nuvem.
- Software como Serviço (*Software as a Service – SaaS*): quando aplicações são disponibilizadas ao usuário final através de navegadores na internet ou via interface de um aplicativo. Neste caso, quem consome o recurso não tem conhecimento sobre a infraestrutura que o suporta e sendo assim, tampouco a gere.

Com relação ao provisionamento, nuvens podem ser: privadas, quando pertencentes a uma organização e, portanto, disponíveis apenas para seus respectivos usuários; públicas, quando seus recursos estão disponíveis para a comunidade em geral; comunitárias, quando seus recursos são provisionados para atender a um grupo específico de usuários que compartilham uma necessidade; ou híbridas, quando compartilham características dos três serviços supracitados.

Por outro lado, Armbrust e colaboradores (Armbrust et al., 2010) não fazem distinção entre os provisionamentos do tipo *PaaS* e *IaaS* e categorizam todos os serviços ofertados em uma nuvem como do tipo *SaaS*, sob a justificativa de que os conceitos de plataforma e infraestrutura sofrem de falta de clareza. Para além disto, citam apenas os conceitos de nuvens públicas, nas quais os serviços são cobrados; e privadas, quando os recursos computacionais pertencem a uma organização e não estão disponíveis para o público em geral. Mais além, definem a computação em nuvem como a junção do serviço *SaaS* com a cobrança pelo uso dos recursos computacionais.

Outro aspecto relevante associado a este ambiente computacional e que representa um dos maiores anseios dos usuários está relacionado à capacidade de aceleração dos experimentos executados.

Conforme a Lei de Amdahl propõe (Amdahl, 1967; Rodgers, 1985), para cada aplicação projetada para executar em um ambiente distribuído, existe um ponto de inflexão, associado à subtarefas intrinsecamente sequenciais.

Especificamente, independentemente da adição de nós computacionais de maneira consecutiva, o ganho de aceleração total tende a ser limitado pelo tempo necessário de processamento de subtarefas que são sequenciais por natureza, acrescido o tempo necessário para processar as demais tarefas (paralelas).

## 1.6 **Genômica Comparativa em Nuvem**

A crescente demanda por capacidade computacional, aliada a necessidade de processar extensos volumes de dados genômicos têm se refletido na criação de soluções para genômica comparativa em ambiente de nuvem computacional.

Este ambiente torna-se ainda mais cativante para as tarefas de genômica comparativa uma vez que muitas aplicações a ela relacionadas combinam etapas que ora requerem uma extensa demanda por capacidade computacional, ora necessitam lidar com grandes volumes de dados, ou ainda utilizam tarefas que demandam menor capacidade em ambos os aspectos supracitados (Shanker, 2012).

O algortimo RSD, desenvolvido por Wall e DeLuca (Wall et al., 2003; Wall & Deluca, 2007) para inferência de ortólogos, tem por base o cálculo de melhores *hits* recíprocos (RBH) através de alinhamentos globais de sequências e a estimativa de distâncias evolutivas por máxima verossimilhança. Os alinhamentos são realizados com a ferramenta Kalign (Lassmann & Sonnhammer, 2005) e as distâncias entre ortólogos através do PAML 4.0 (Yang, 2007). Os grupos ortólogos são construídos de tal forma que: cada gene de um grupo é ortólogo a ao menos outro gene do mesmo grupo, porém não é ortólogo a nenhum gene de nenhum outro grupo. Esta abordagem é denominada agrupamento determinístico por ligação única (*deterministic single-linkage clustering*). Este algoritmo para inferência de ortólogos foi implantado com sucesso na nuvem EC2 da Amazon (Amazon, 2012; Wall et al., 2010) em 100 nós computacionais através da tecnologia MapReduce (Dean & Ghemawat, 2008). Os agrupamentos de ortólogos obtidos foram incorporados à base de ortólogos Roundup (DeLuca et al., 2012; Wall et al., 2010), que atualmente conta com mais de 1800 genomas.

CloudBlast (Matsunaga et al., 2008) oferece a possibilidade de executar a aplicação BLAST (Altschul et al., 1990) em nuvem através da fragmentação dos

dados de entrada – sequências – e o processamento dos mesmos em nós computacionais distribuídos, também através da tecnologia MapReduce.

Por fim, a aplicação Crossbow (Gurtowski et al., 2012; Langmead et al., 2009a) é mais um exemplo de aplicação de genômica comparativa e tem por objetivo identificar polimorfismos de nucleotídeos únicos (*SNP's*). Utiliza a aplicação Bowtie (Langmead et al., 2009b) para alinhamento das sequências a serem analisadas, em conjunto com o pacote SOAPsnp (Li et al., 2009) para identificação de SNP's. Foi construído com o suporte da linguagem de programação Java, também emprega o uso da metodologia MapReduce e foi avaliado sob a nuvem EC2 da Amazon.

## 1.7 **Organização da Tese**

Esta tese possui como tema central o estudo da genômica comparativa em ambiente computacional distribuído e é resultado de trabalho colaborativo entre o LBCS/IOC/Fiocruz e do I3M/UPV, durante o período de Doutorado Sanduíche em bolsa[1] concedida pelo PDSE/CAPES (CAPES, 2015) para o autor desta tese.

Nesta tese também foi desenvolvida contribuição relevante para a genômica comparativa: uma nova metodologia para a criação de bases de grupos ortólogos aprimoradas através do OrthoSearch.

Todos os resultados obtidos nesta tese são explorados nos capítulos a seguir. O Capítulo 3 apresenta o material e métodos utilizados no desenvolvimento destas soluções. Detalhadamente:

- O Capítulo 3.1 descreve o produto principal desta tese, o *elastic-OrthoSearch*. Este é um novo *workflow* científico de genômica comparativa para inferência de homologia entre organismos, desenvolvido em parceria com o *Instituto de Instrumentación para Imagen Molecular (I3M)* da *Universitat Politècnica de València* (UPV) e avaliado em ambiente de nuvem computacional.

[1] Bolsa PDSE/CAPES BEX 2137/14-3

- O Capítulo 3.2 apresenta as atualizações realizadas no OrthoSearch e a metodologia proposta para a criação de bases de ortólogos aprimoradas a partir do OrthoSearch.

O Capítulo 4 e seus sub-tópicos apresentam os resultados obtidos nestas atividades. O Capítulo 5 é composto pela discussão dos resultados alcançados e o Capítulo 6 apresenta as conclusões sobre esta tese.

Destacam-se também os Anexos 8.1, 8.2, 8.3, 8.4 e 8.5 que representam cinco manuscritos, dois submetidos e um publicado (em revistas internacionais) e dois publicados em congressos internacionais, a saber:

- Anexo 8.1: "*Design and implementation of a Generic and Multi-Platform Workflow System*" – publicado no congresso "8th Ibergrid Conference (*Iberian Grid Infrastructure Conference*)", em Aveiro, Portugal, em setembro de 2014. Este manuscrito representa os resultados da primeira interação entre o autor da tese e a equipe da UPV para entendimento do problema a ser solucionado;
- Anexo 8.2: "*elastic-OrthoSearch: a cloud-based comparative genomics workflow*" – este manuscrito versa sobre a criação de uma solução para inferência de ortólogos em nuvem computacional (elastic-OrthoSearch), a partir do OrthoSearch e foi submetido para a revista *Evolutionary Bioinformatics*. No momento, está em revisão pela mesma;
- Anexo 8.3: "*Improved orthologous databases to ease Protozoan targets inference*" – publicado na revista *Parasites & Vectors*. Associa-se à nova metodologia proposta pelo autor, em colaboração com os pesquisadores Dr. Alberto Dávila e Dr. Rodrigo Jardim para a criação de novas bases de ortólogos a partir do OrthoSearch.
- Anexo 8.4: **"***A Multi-Platform Workow Management System for supporting e-Science experiments***"** – submetido à revista *Journal of Systems and Software*, detalha a máquina de *workflow* criada pela equipe do I3M/UPV durante o período de doutorado sanduíche no exterior do autor da tese.
- Anexo 8.5: "A Multi-Platform Workflow Management System optimized for Cloud Computing Platforms" – publicado na conferência "21st

International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA'15)" em Las Vegas, Estados Unidos da América. Este manuscrito aborda um experimento realizado no I3M/UPV com o uso do elastic-OrthoSearch em ambiente computacional híbrido (nuvem e *cluster*).

# 2 OBJETIVOS

## 2.1 Objetivo Geral

Propor uma solução de genômica comparativa para inferência de ortólogos, através de plataforma tecnológica distribuída e paralela.

## 2.2 Objetivos Específicos

### *2.2.1 Estabelecer uma nuvem piloto do tipo privada no Laboratório de Biologia Computacional e Sistemas do Instituto Oswaldo Cruz (LBCS/IOC).*

### *2.2.2 Desenvolver um workflow científico completo de genômica comparativa para inferência de ortólogos na nuvem (baseado no OrthoSearch).*

### *2.2.3 Adotar estratégia de programação paralela e/ou distribuída para a nova proposta.*

### *2.2.4 Oferecer uma nova metodologia para criação de bases de ortólogos aprimoradas a partir do OrthoSearch.*

# 3 MATERIAL E MÉTODOS

## 3.1 elastic-OrthoSearch e a nuvem computacional

### 3.1.1 *Caracterização e processo de criação da proposta de solução*

Esta é a contribuição principal desta tese, diretamente associada aos objetivos específicos 2.2.1, 2.2.2 e 2.2.3. Seu produto é denominado elastic-OrthoSearch, um *workflow* científico de genômica comparativa executável em ambiente computacional distribuído – a nuvem.

A realização desta contribuição ocorreu em duas etapas. A primeira, realizada exclusivamente pelo autor da tese, no LBCS/IOC/Fiocruz, no Brasil, está associada aos objetivos específicos 2.2.1 e 2.2.2.

Especificamente, a plataforma de nuvem OpenNebula (OpenNebula, 2015) foi selecionada, após experimentos com Eucalyptus (Nurmi et al., 2009), preterida à época por não oferecer suporte à distribuição Linux CentOS (CentOS, 2015), utilizada por alguns de nossos servidores que compõem a nuvem; e OpenStack (OpenStack, 2010), preterida por apresentar complexidade de instalação e configuração muito superior ao OpenNebula, sem oferecer vantagens ou diferenciais relevantes para os problemas abordados nesta tese.

As atividades da segunda etapa desta contribuição, referentes ao objetivo específico 2.2.3 foram realizadas na Espanha, durante o período de doutorado sanduíche no exterior. Foram coordenadas pelo Dr. Ignacio Blanquer-Espert (I3M/UPV), que atribuiu ao doutorando em Informática Abel Carrión, membro de seu laboratório, a responsabilidade de desenvolver o elastic-OrthoSearch junto ao autor desta tese.

A equipe do I3M/UPV teve por responsabilidade: desenvolver uma máquina de *workflow* que suportasse a execução do elastic-OrthoSearch na nuvem, parte componente da tese do discente do I3M/UPV supracitado; trabalhar na gestão da infraestrutura computacional da nuvem (criação, manipulação e destruição dos nós computacionais); criar mecanismos para registrar as atividades de cada estágio de execução do elastic-OrthoSearch e prover mecanismos de elasticidade ao mesmo, tais como o acréscimo e/ou liberação de recursos computacionais de cada experimento, que até o momento ainda são definidos previamente à execução dos experimentos.

O autor desta tese atuou na instalação e configuração da nuvem computacional do LBCS/IOC/Fiocruz, na definição dos requisitos para a máquina de *workflow*, na concepção do *workflow* do elastic-OrthoSearch, no levantamento da interdependência e interligação das tarefas do fluxo, na definição das possibilidades de distribuição e paralelismo das mesmas, na montagem, realização e análise dos experimentos, na seleção das bases de dados e organismos, na atualização e preparação do OrthoSearch para executar na nuvem.

Foram selecionados e preparados os dados de entrada (base de ortólogos e proteomas de organismos) e foram montados cenários em uma nuvem privada baseada em *OpenNebula* (OpenNebula, 2015), padrão de nuvem definida no LBCS/IOC/FIOCRUZ, devido à sua maturidade e estabilidade no processo de instalação e configuração de recursos computacionais.

Ao todo, foram realizados 15 experimentos e os dados de aceleração e quantidade de ortólogos obtidos foram compilados e analisados pelo autor da tese.

### 3.1.2 *Projeto do elastic-OrthoSearch*

O elastic-OrthoSearch é um *workflow* científico de genômica comparativa para inferência de ortólogos em uma estratégia de identificação de melhores *hits* recíprocos com suporte à construção e utilização de perfis de cadeias de Markov.

Nossa solução foi criada para manter ao máximo o desenho original do OrthoSearch. A Figura 3.1 apresenta as etapas descritas (da Cruz et al., 2008) para o OrthoSearch em sua versão original (da Cruz et al., 2010).

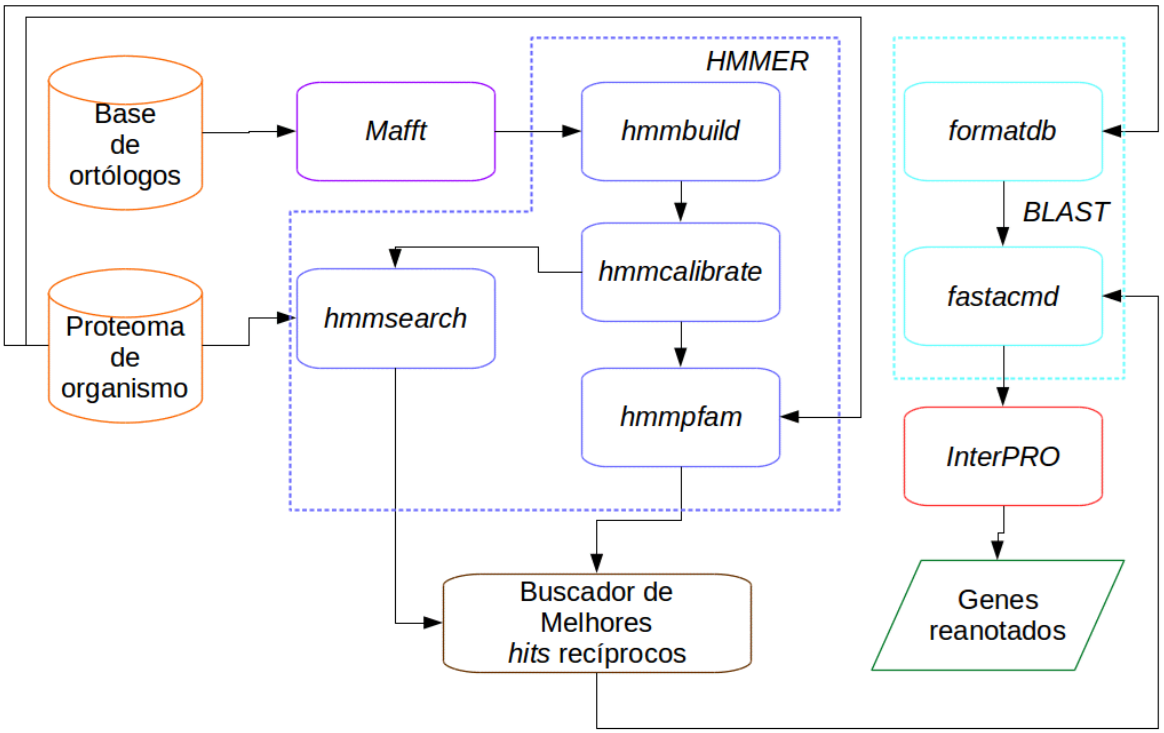


Figura 3.1: *Workflow* abstrato do OrthoSearch em sua versão original. (Fonte: adaptado de (da Cruz et al., 2008)).

Nesta tese, anteriormente ao projeto do elastic-OrthoSearch, atualizamos o OrthoSearch para contemplar o uso dos aplicativos da suíte HMMER na versão 3. Além disso, desacoplamos o OrthoSearch de sistemas gerenciadores de *workflows* e adotamos as linguagens de programação C++ e Ruby para as chamadas de execução de seus estágios.

A Figura 3.2 apresenta o *workflow* abstrato do elastic-OrthoSearch com todas as suas atividades, dados de entrada e saída.

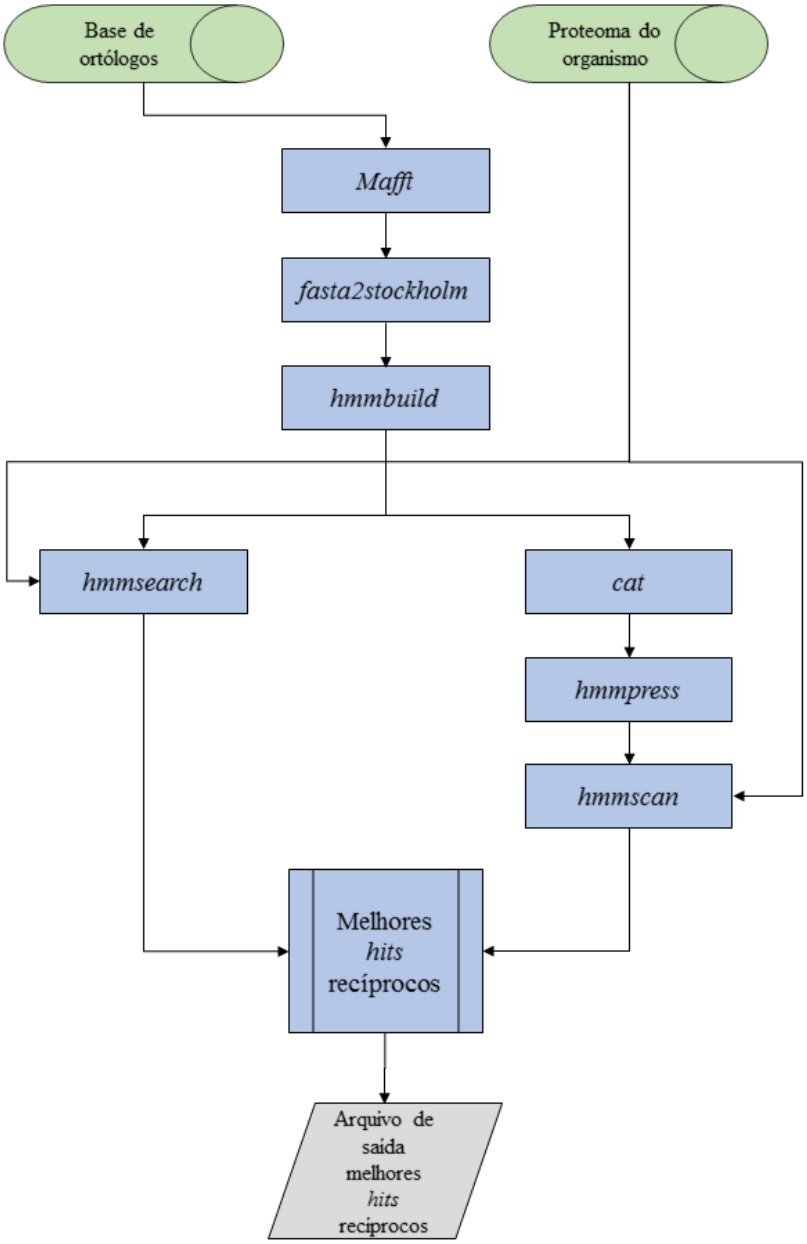


Figura 3.2: *Workflow* abstrato do elastic-OrthoSearch.

A máquina de *workflow* desenvolvida no I3M/UPV (Carrión et al., 2014) conta com uma etapa de tradução destas atividades em estágios, que especificam as ações de solicitação de criação de instâncias (nós computacionais), a cópia de dados necessários para a realização de cada tarefa, a entrega de resultados intermediários de execução e a entrega do resultado final ao usuário.

Estas atividades foram traduzidas para a criação de um *workflow* concreto na nuvem, conforme apresentado na Figura 3.3. Para tal, foram criados cinco estágios

para o elastic-OrthoSearch, a saber: “mafft-hmmbuild”; “hmmsearch”; “cat”; “hmmpress-hmmscan” e “best-hits”.

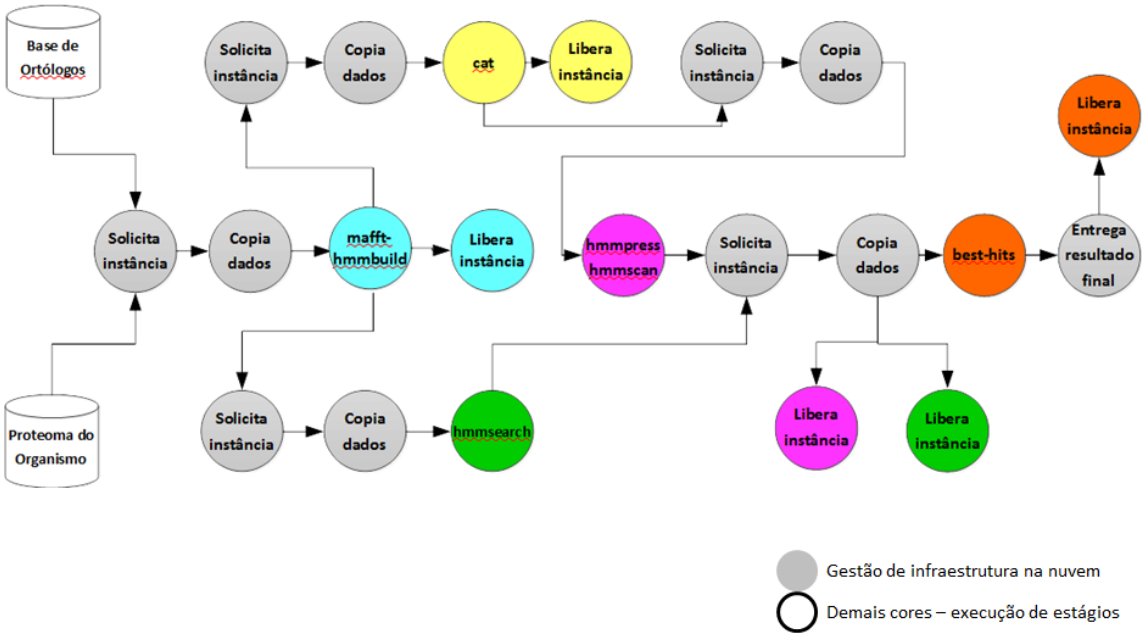


Figura 3.3: *Workflow* concreto do elastic-OrthoSearch

A tarefa “mafft-hmmbuild” realiza o alinhamento dos grupos ortólogos da base de entrada com o algoritmo Mafft (Katoh & Standley, 2013), converte os resultados em formato Stockholm (Sonnhammer, 2015) (para melhor compatibilidade com a suíte HMMER (Finn et al., 2011)) através de um *script* em linguagem Perl desenvolvido no LBCS/IOC/FIOCRUZ (Anexo 8.6) e constrói os perfis HMM através do aplicativo *hmmbuild*.

Em seguida, a tarefa “hmmsearch” confronta os perfis HMM, um a um, contra o proteoma do organismo selecionado como dado de entrada, e gera um arquivo de saída para cada perfil confrontado, que contém uma lista ordenada de melhores *hits* do proteoma para com o mesmo.

Posteriormente, a tarefa “cat” é responsável por concatenar os perfis HMM construídos pelo estágio “hmmbuild” em um único arquivo texto (base de perfis HMM). Esta atividade é pré-requisito para a tarefa “hmmpress-hmmscan”, que comprime esta base de perfis HMM via hmmpress e em seguida a processa pelo algortimo hmmscan, que realiza a busca de melhores *hits* confrontando cada proteína do organismo contra esta base de perfis HMM e gera uma lista ordenada de *hits*.

As duas listas, geradas pelos estágios “hmmsearch” e “hmmpress-hmmscan”, são confrontadas em algoritmo escrito pelo autor da tese, em linguagem Ruby, que corresponde a tarefa “best-hits” (Anexo 8.7).

Esta infere os grupos ortólogos da base de ortólogos que correspondem aos melhores *hits* recíprocos entre a mesma e o organismo selecionado. Como a lista do hmmsearch corresponde aos melhores *hits* entre perfis HMM e proteoma e a lista do hmmscan aos *hits* entre proteoma e perfis concatenados, temos um confronto bidirecional, doravante temos melhores *hits* recíprocos, entregues como um arquivo em formato CSV (Anexo 8.8).

Dentre todas estas tarefas, aquela que representa o maior custo computacional e temporal em todo o *workflow* é a tarefa “mafft-hmmbuild”, seguida pela tarefa “hmmpress-hmmscan”.

Adicionalmente, apenas as tarefas “mafft-hmmbuild” e “hmmsearch” são suscetíveis à paralelização/distribuição, uma vez que os dados de entrada das mesmas podem ser distribuídos, fragmentados entre distintos recursos computacionais, sem perda de informação biológica.

A execução das tarefas do elastic-OrthoSearch ocorre em nós computacionais na forma de máquinas virtuais, denominadas instâncias. Estas instâncias são criadas, configuradas, gerenciadas e destruídas na nuvem, através da máquina de *workflow* desenvolvida por esta tese no I3M/UPV (Carrión et al., 2014) que por sua vez é integrada ao componente *Infrastructure Manager* (IM) (Caballer et al., 2015), também desenvolvido no I3M/UPV e responsável por gerenciar a infraestrutura de instâncias criadas na nuvem.

A máquina de *workflow* é responsável por oferecer a elasticidade ao elastic-OrthoSearch, previamente à execução do mesmo, através de um arquivo de configuração no formato JSON (ECMA, 2013), no qual o usuário informa as características de *hardware* e *software* para cada estágio do elastic-OrthoSearch, conforme no Anexo 8.9.

Até o momento, o registro de atividades do elastic-OrthoSearch está limitado à guarda e sinalização de execução de cada tarefa do *workflow*. Essas informações persistem apenas no escopo do experimento, em uma base suportada pelo SGBD MongoDB (MongoDB, 2015) para acompanhamento da execução do mesmo e são armazenadas em um arquivo de *log* de execução, exemplificado no Anexo 8.10.

### 3.1.3 *Dados de entrada*

Assim como o OrthoSearch, o elastic-OrthoSearch requer uma base de ortólogos e os dados de proteínas de um organismo em formato *multifasta* (proteoma) como dados de entrada. Para a realização dos experimentos com o elastic-OrthoSearch, selecionamos a base EggNOG em sua versão 4.0 (Powell et al., 2013).

Devido única e exclusivamente a limitações de recursos computacionais disponíveis, que impossibilitariam a execução dos experimentos desejados, selecionamos um dos subconjuntos da base eggNOG, disponível no sítio da mesma (*http://eggnog.embl.de*), que conta exclusivamente com agrupamentos não supervisionados de eucariotos (subconjunto euNOG). Este subconjunto conta com até duas proteínas de cada organismo em cada um dos grupos ortólogos disponíveis, selecionadas de forma aleatória. A Tabela 3.1 detalha os dados da base EggNOG e do subconjunto euNOG utilizado para os nossos experimentos.

Tabela 3.1: Detalhes da base EggNOG 4.0 e do subconjunto EggNOG euNOG utilizado nos experimentos do elastic-OrthoSearch.

| **Base de dados** | **Organismos** | **Grupos de ortólogos** | **Total de proteínas** | **Tamanho da base** |
|---|---|---|---|---|
| EggNOG 4.0 | 3.686 | >1.700.000 | >7.700.000 | 1.9 GBytes |
| EggNOG euNOG | 282 | 60.164 | 2.183.224 | 1.4 GBytes |

Os organismos selecionados para a realização dos experimentos foram: *Cryptosporidium hominis*, *Entamoeba histolytica* e *Leishmania infantum*. A seleção foi aleatória dentre os protozoários disponíveis no ProtozoaDB (BiowebDB, 2012; Dávila et al., 2008), de onde foram recuperados proteomas de cada uma das espécies.

A Tabela 3.2 apresenta os dados de cada um dos proteomas selecionados para os experimentos, com o total de proteínas para cada organismo e o tamanho de seu proteoma em mega pares de bases (Mpb).

Tabela 3.2: Detalhes do proteoma de cada organismo utilizado nos experimentos do elastic-OrthoSearch.

| Organismo | Total de proteínas | Tamanho (Mpb) |
|---|---|---|
| *Cryptosporidium hominis* | 3.885 | 9,16 |
| *Entamoeba histolytica* | 7.973 | 24 |
| *Leishmania infantum* | 7.872 | 32,13 |

### 3.1.4 *Cenários para execução dos experimentos*

Os experimentos foram realizados em duas configurações distintas: uma com a execução do OrthoSearch em sua forma serializada, não elástica; e outra, elástica, com o *elastic-OrthoSearch*. Para cada configuração, foram montados cenários com os três organismos selecionados e a base de ortólogos EggNOG 4.0.

A execução não elástica ocorreu em cenário com uma única máquina com sistema operacional Ubuntu 14.04 com 16 núcleos, 24GB de memória e 100GB de disco rígido local.

A execução elástica (elastic-OrthoSearch) na nuvem ocorreu em quatro cenários distintos, para cada organismo: com duas, quatro, oito e 16 instâncias computacionais na forma de máquinas virtuais, em uma nuvem OpenNebula 4.8.

Com as três execuções no cenário não elástico (uma para cada organismo) e as 12 demais execuções elásticas (elastic-OrthoSearch) na nuvem, foram realizados 15 experimentos, conforme apresentado pela Tabela 3.3.

Tabela 3.3: Cenários de experimentos realizados para cada organismo e tipo de cenário (elástico/não-elástico), com o total de experimentos.

| **Organismo/Cenário** | **Não elástico (OrthoSearch)** | **Elástico (elastic-OrthoSearch)** |
|---|---|---|
| *Cryptosporidium hominis* | 1 instância | 2, 4, 8 e 16 instâncias |
| *Entamoeba histolytica* | 1 instância | 2, 4, 8 e 16 instâncias |
| *Leishmania infantum* | 1 instância | 2, 4, 8 e 16 instâncias |
| **Execuções por cenário** | **3** | **12** |
| **Total de execuções** | **15** | |

Para cada estágio da execução do *workflow* foi realizada configuração distinta de *hardware*, de acordo com as caraterísticas de cada aplicação associada (maior demanda por recursos de memória e/ou armazenamento), conforme listado na Tabela 3.4.

Tabela 3.4: Detalhes da configuração de *hardware* de cada instância para a execução do elastic-OrthoSearch, por estágio envolvido. Um experimento corresponde pela execução de todos os estágios.

| ***Hardware*/Estágio** | **"mafft-hmmbuild"** | **"hmmsearch"** | **"cat"** | **"hmmpress-hmmscan"** | **"best-hits"** |
|---|---|---|---|---|---|
| Núcleos de CPU (*cores*) | 1 | 1 | 1 | 1 | 1 |
| Memória RAM | 4 GBytes | 4 Gbytes | 4 GBytes | 24 GBytes | 24 GBytes |
| Espaço em disco (local) | - | - | 500GB | 500GB | 50GB |
| Espaço em disco (*NFS*) | 40 GBytes | 40 GBytes | - | - | - |

## 3.2 **Metodologia para a criação de bases de ortólogos aprimoradas**

### **3.2.1** ***Melhorias no OrthoSearch e cenários para os experimentos***

A segunda contribuição desta tese, associada ao objetivo específico 2.2.4 é a proposta de uma metodologia capaz de criar bases de ortólogos aprimoradas através do OrthoSearch.

Todas as atividades desta contribuição foram desenvolvidas no LBCS/IOC/Fiocruz, no Brasil, à exceção da redação do artigo associado à mesma, que ocorreu de forma remota e concorrente ao período de doutorado sanduíche no exterior, na Espanha.

Inicialmente, as rotinas que compõem o OrthoSearch foram atualizadas para adotar o (i) HMMER versão 3 e (ii) para utilizarem as linguagens de programação C++ 4.43 e Ruby 1.8.7, ao invés de Perl.

Estas atualizações foram realizadas para aproveitar o ganho de desempenho fornecido pelo algoritmo heurístico de aceleração adotado no HMMER3 em relação às versões anteriores. Adicionalmente, as atualizações dos algoritmos *hmmbuild*, *hmmsearch* e *hmmscan* (anteriormente denominado *hmmpfam*) permitem ao usuário a possibilidade de adotar parâmetros para multiprocessamento (aproveitamento de todos os núcleos de processamento disponíveis), dentre outros.

Todos os experimentos foram realizados em um único servidor com sistema operacional Ubuntu 12.04, com 64 núcleos de processamento (*cores*), 32 GBytes de memória RAM e 500 GBytes de espaço em disco.

### 3.2.2 *Inferência de ortólogos com o OrthoSearch*

O OrthoSearch requer dois tipos de dados de entrada: (i) uma base de ortólogos – arquivos em formato *multifasta* e (ii) um arquivo em formato *multifasta* com as proteínas de um organismo. Previamente à criação das bases de ortólogos, foram realizados experimentos para inferência de ortólogos para três protozoários e três bases de ortólogos, em total de nove experimentos. Estes serviram como base comparativa para a análise dos resultados obtidos com a criação das novas bases de ortólogos.

As bases utilizadas foram: Kegg Orthology (KO) (Kanehisa et al., 2012; Kegg Orthology, 2012), EggNOG KOG (Powell et al., 2011) e ProtozoaDB (Dávila et al., 2008). A base KO foi obtida através de FTP e contém proteínas de organismos de toda a árvore da vida – *Archaea*, *Bacteria* e *Eukarya*. EggNOG KOG é um subconjunto da base EggNOG, contém apenas proteínas de eucariotos e foi obtida através do sítio da mesma. ProtozoaDB é uma base de ortólogos composta de proteínas de 22 protozoários. Os detalhes sobre cada base de ortólogos citada podem ser visualizados na Tabela 3.5.

Tabela 3.5: Características de cada base utilizada para inferência de ortólogos com o OrthoSearch.

| **Base de ortólogos** | **Organismos** | **Protozoários** | **Total de grupos ortólogos** | **Total de proteínas** | **Tamanho da base** |
|---|---|---|---|---|---|
| KO | 1.535 | 35 | 14.856 | 2.108.653 | 1,2 GBytes |
| EggNOG KOG | 55 | 15 | 4.851 | 409.796 | 267 MBytes |
| ProtozoaDB | 22 | 22 | 13.792 | 64.182 | 128 MBytes |

Foram selecionados, três espécies de protozoários para serem confrontadas junto às três bases de ortólogos supracitadas, a saber: *Cryptosporidium hominis*, *Entamoeba histolytica* e *Leishmania infantum*, presentes em distintos clados da árvore da vida.

Os arquivos com os dados de proteínas em formato *multifasta* das mesmas foram recuperados do ProtozoaDB. As características de cada proteoma são descritas na Tabela 3.6.

Tabela 3.6: Detalhes sobre os proteomas de cada organismo utilizado nos experimentos com o OrthoSearch.

| **Organismo** | **Total de proteínas** | **Tamanho em pares de bases** | **Tamanho em disco** |
|---|---|---|---|
| *Cryptosporidium hominis* | 3.885 | 9,1 Mpb | 2,1 MBytes |
| *Entamoeba histolytica* | 7.973 | 24 Mpb | 4,0 MBytes |
| *Leishmania infantum* | 7.872 | 32,13 Mpb | 5,4 MBytes |

### 3.2.3 *Uso do OrthoSearch para a construção de bases de ortólogos*

A construção de novas bases de ortólogos através do OrthoSearch, uma nova metodologia não incialmente prevista como o objetivo desta tese, ganhou relevância após a realização de experimentos preliminares, ainda com o OrthoSearch.

Com isto obtivemos subsídios para investigar e aprimorá-la, em atividades de pesquisa que convergiram para a publicação de artigo em revista internacional indexada (Kotowski et al., 2015).

Através da metodologia proposta nesta tese, o OrthoSearch passa a oferecer, além da (i) inferência de ortólogos, a possibilidade de (ii) criar novas bases de ortólogos, tal como apresentado na Figura 3.4.

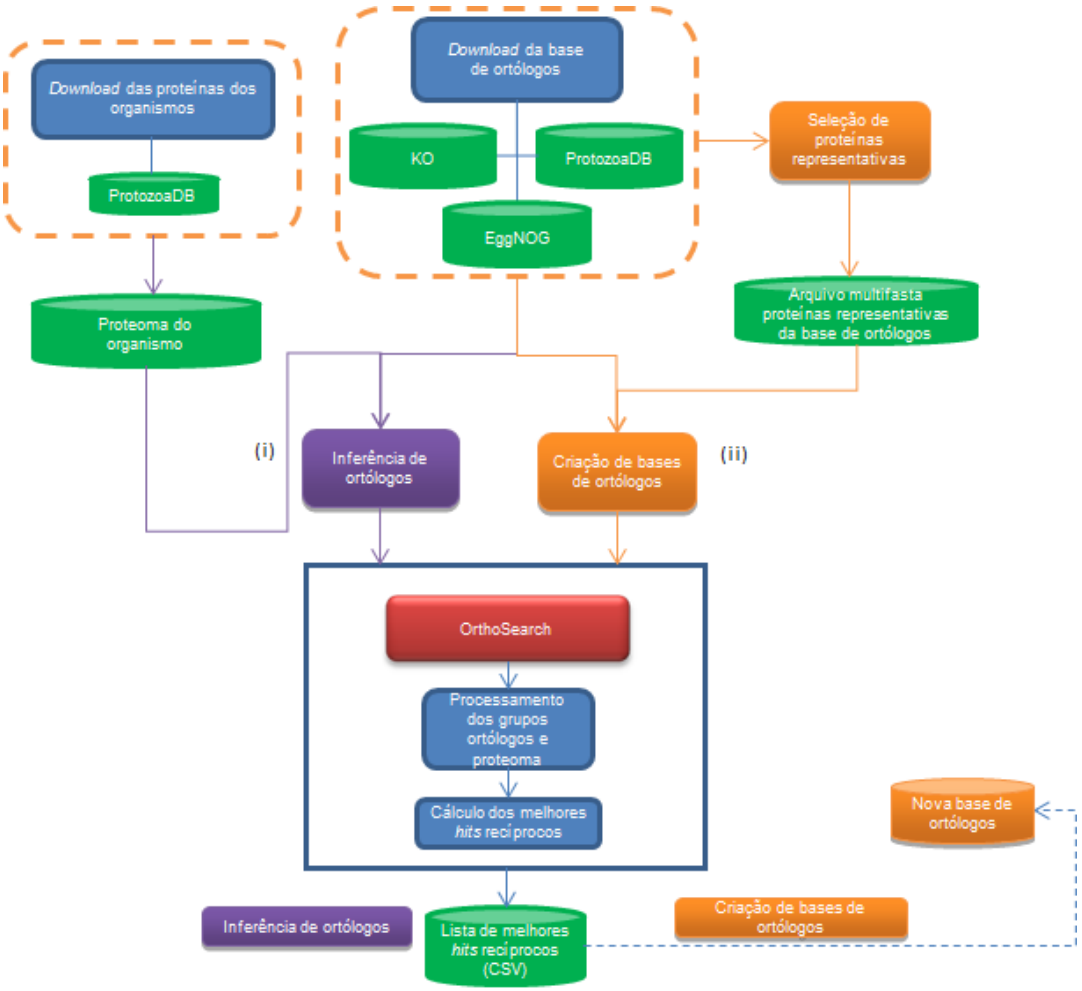


Figura 3.4: As duas opções para utilização do OrthoSearch: (i) inferência de ortólogos e (ii) criação de novas bases de ortólogos.

A criação das novas bases utiliza requereu a adaptação dos dados de entrada do OrthoSearch. Em uma execução tradicional, o OrthoSearch espera confrontar uma base de ortólogos contra um proteoma de um organismo, que é um arquivo com suas proteínas, em formato *multifasta*.

Para a criação de uma nova base, mantivemos uma base de ortólogos como um dos dados de entrada e substituímos o proteoma de um organismo por um único arquivo *multifasta* que contém proteínas representativas de grupos ortólogos de outra base, como um proteoma simulado.

Estas proteínas representativas de cada grupo ortólogo foram selecionadas com base em um *script* fornecido pelo Dr. Salvador Capella (Capella-Gutiérrez, 2014). O *script* utiliza como base o algoritmo trimAl (Capella-Gutierrez et al., 2009), que recebe como entrada os alinhamentos múltiplos de sequências de grupos de ortólogos.

Para cada alinhamento, de cada grupo ortólogo, inicialmente, as regiões de baixa qualidade são removidas (*gaps*) e em seguida calcula-se, par a par, dentre os alinhamentos deste grupo ortólogo, a sequência que possui o maior *score* de identidade, ou seja, a maior quantidade de resíduos idênticos. Esta é então selecionada como a proteína representativa dentre as que compõem o grupo ortólogo.

Essa seleção é realizada para todos os grupos ortólogos da base, que ao final é consolidada como um proteoma simulado, composto por cada uma das proteínas representativas de cada um de seus grupos ortólogos, concatenadas em um único arquivo *multifasta* através de outro *script*, desenvolvido pelo autor da tese, em linguagem Ruby. Foram armazenados o identificador e a sequência de aminoácidos de cada uma das proteínas, conforme apresentado no Anexo 8.11.

Em um processo de retroalimentação, iterativo, selecionamos como ponto de partida a base que com a maior representação de organismos de toda a árvore da vida, KO, o que em teoria representa maior oferta de variabilidade de proteínas. Neste momento, ao invés de confrontá-la frente um proteoma de organismo, utilizamos o arquivo *multifasta* com as proteínas representativas de EggNOG KOG.

Como em uma execução tradicional do OrthoSearch, os melhores *hits* recíprocos foram processados com o apoio de *scripts* internos desenvolvidos em Ruby e *shell scripts* Unix/POSIX, que montam os grupos ortólogos da nova base“KO + EggNOG KOG” de acordo com as regras citadas a seguir.

Os grupos ortólogos desta base são: (i) grupos originais de KO que não apresentaram melhores *hits* recíprocos com as proteínas representativas (portanto, com nenhum grupo ortólogo) de EggNOG KOG; (ii) grupos originais de EggNOG KOG cujas proteínas representativas não obtiveram *hit* contra KO; e (iii) grupos de KO que obtiveram melhor *hit* recíproco com proteínas representativas de EggNOG KOG e foram atualizados para conter todas as proteínas do grupo ortólogo em EggNOG KOG de onde a proteína representativa tem sua origem.

Esta mesma sequência de passos foi utilizada para criar a segunda base de ortólogos desta tese, através do confronto entre “KO + EggNOG KOG” e as proteínas representativas de ProtozoaDB, o que gerou a base “KO + EggNOG KOG + ProtozoaDB”.

### 3.2.4 *Inferência de ortólogos com as novas bases construídas*

Com as duas bases de ortólogos criadas pela metodologia proposta, “KO + EggNOG KOG” e “KO + EggNOG KOG + ProtozoaDB”, foi realizada nova série de experimentos com o OrthoSearch contra os três protozoários supracitados.

### 3.2.5 *Comparação entre as novas bases e o OrthoMCLDB*

Também foram realizados experimentos de inferência de ortólogos entre os três protozoários através da aplicação *web* do OrthoMCLDB. Os resultados obtidos foram tabulados sob ponto de vista quantitativo entre o OrthoMCLDB e as bases geradas pela metodologia proposta para posterior análise.

### 3.2.6 *Alvos potenciais em Leishmania spp. contra o protema humano*

Em outra aplicação prática para a metodologia proposta nesta tese, procuramos identificar alvos potenciais de fármacos para *Leishmania* spp., protozoário de interesse de pesquisa do LBCS/IOC.

Foi realizado um BlastP (Camacho et al., 2009) entre a base “KO + EggNOG KOG + ProtozoaDB” (uma amostra da listagem com os grupos ortólogos e proteínas que obtiveram *hit* com o proteoma humano estão disponíveis no Anexo 8.12) e o proteoma humano, obtido através do RefSeq (NCBI, 2015).

Foi utilizado o BlastP versão 2.2.28+ com *e-value* 0.1, para maximizar a estringência das sequências analisadas para com o proteoma humano e por consequência ser possível capturar homólogos mais distantes. Os grupos ortólogos que não obtiveram *hit* contra o proteoma humano, mas que o tiveram contra *Leishmania* spp. foram analisados pois poderiam conter alvos para este gênero.

Também foi executado um BlastP contra as bases KO, EggNOG KOG e ProtozoaDB, separadamente.

# 4 RESULTADOS

## 4.1 elastic-OrthoSearch e a nuvem computacional

O elastic-OrthoSearch foi avaliado em total de 15 experimentos de inferência de ortólogos; os dados de aceleração e quantidade de ortólogos inferidos foram compilados e analisados, para cada organismo selecionado.

Foi utilizada a base de ortólogos EggNOG euNOG, contra os organismos *Cryptosporidium hominis*, *Entamoeba histolytica* e *Leishmania infantum*.

Os resultados do elastic-OrthoSearch são entregues na forma de um arquivo separado por vírgulas (CSV), que contém as informações referentes aos melhores *hits* recíprocos identificados, tal como é possível observar no Anexo 8.8. A primeira linha do arquivo é o cabeçalho e informa desde o grupo ortólogo identificado como melhor hit recíproco (*BEST_MODEL_QUERY*) até a descrição da proteína do organismo que apresentou este melhor *hit* recíproco (*HMMSEARCH_BEST_DESCRIPTION*), além de todos os valores informados pelos aplicativos *hmmsearch* e *hmmscan*.

As etapas de bioinformática (aqui incluídos os alinhamentos múltiplos dos grupos ortólogos com o Mafft, a construção e manipulação de perfis HMM via HMMER3 e a identificação de melhores *hits* recíprocos) foram analisadas sob o ponto de vista de eligibilidade a paralelismo e/ou distribuição de tarefas.

A natureza do *workflow* e das aplicações de bioinformática utilizadas determinaram os limites para a possibilidade de distribuição e paralelismo das tarefas a serem executadas, uma vez que tampouco foi escopo desta tese a modificação de algoritmos de tais aplicações. A Figura 4.1 demonstra o processo de distribuição e paralelismo das tarefas.

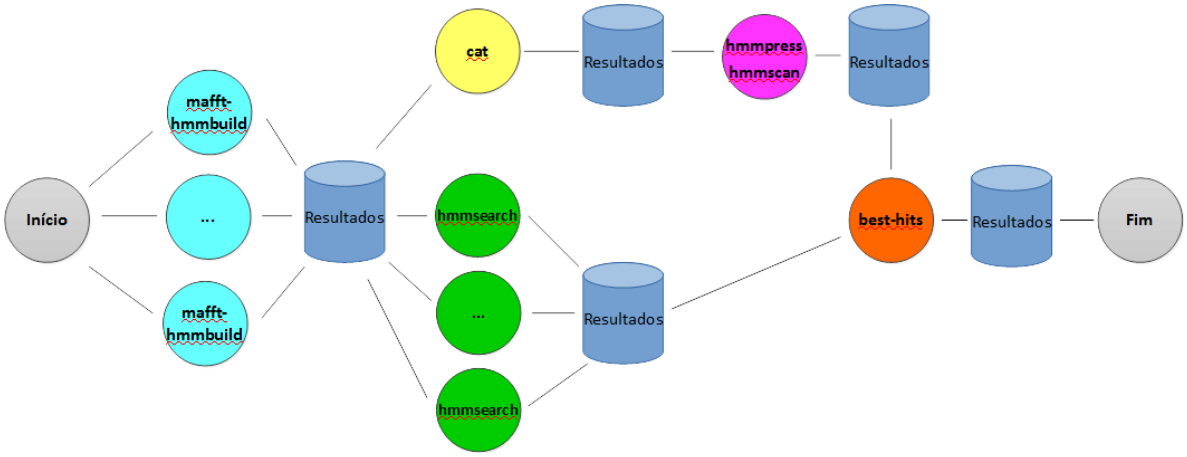


Figura 4.1: Visão do *workflow* científico elastic-OrthoSearch com ênfase nas possibilidades de distribuição e paralelismo de tarefas.

As etapas de “mafft-hmmbuild” e “hmmsearch” utilizaram como armazenamento o recurso de *NFS*. São tarefas que, apesar de realizarem operações de leitura e escrita em uma ordem de arquivos considerável, tais arquivos manipulados são os de menor volume do fluxo, o que não justifica a cópia/replicação de dados para cada instância.

As tarefas “cat”, “hmmpress-hmmscan” e “best-hits” utilizaram armazenamento local. Realizar a cópia local dos dados do *NFS* para as instâncias representa menor penalização na execução do fluxo, pois além dos dados utilizados estarem dentre os maiores volumes manipulados, a natureza das operações envolvidas requer acesso de leitura e escrita em arquivos de forma intensa – o que consumiria largura de banda e poderia influenciar negativamente no tempo para manipular tais arquivos. Além disso, os algoritmos das aplicações supracitadas não suportam paralelização/fragmentação dos dados de entrada.

Com os dados fornecidos pelo arquivo que controla a execução do fluxo e armazena as informações de proveniência de cada estágio foi possível quantificar a aceleração obtida através do elastic-OrthoSearch em comparação à execução fora do ambiente de nuvem (sequencial). A Tabela 4.1 lista os resultados obtidos.

Tabela 4.1: Tempo de execução para cada experimento realizado na nuvem com o elastic-OrthoSearch.

| | **Tempo total de execução (HH:MM) / Aceleração obtida** | | | | | |
|---|---|---|---|---|---|---|
| **Instâncias** | ***Cryptosporidium hominis*** | | ***Entamoeba histolytica*** | | ***Leishmania infantum*** | |
| 1 | 34:10 | N/A | 39:07 | N/A | 39:34 | N/A |
| 2 | 16:36 | 2,122 | 19:35 | 1,997 | 19:52 | 1,992 |
| 4 | 10:43 | 3,188 | 14:06 | 2,774 | 14:27 | 2,739 |
| 8 | 08:20 | 4,122 | 11:31 | 3,396 | 10:52 | 3,642 |
| 16 | 06:54 | 4,951 | 10:11 | 3,841 | 09:35 | 4,130 |

O melhor resultado obtido ocorreu em cenário com 16 instâncias, durante experimento com o organismo *Cryptosporidium hominis*, com aceleração de 4,95 vezes, em tempo total de 06 horas e 54 minutos, contra 34 horas e 10 minutos na execução sequencial.

O pior cenário obtido foi com o organismo *Leishmania infantum* em execução com duas instâncias e mesmo assim, obtivemos aceleração de 1,99 vezes o resultado sequencial. A Figura 4.2 apresenta a curva de aceleração obtida para os experimentos realizados.

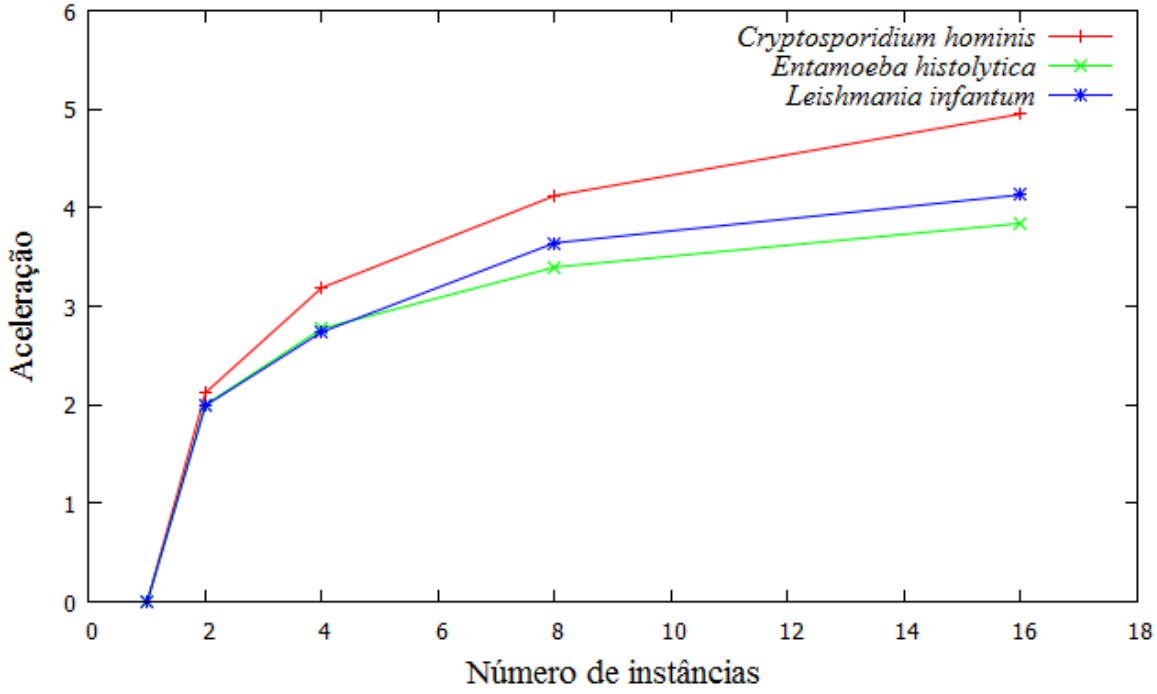


Figura 4.2: Curva de aceleração obtida com o elastic-OrthoSearch com a base EggNOG euNOG, em relação à execução não-elástica (sequencial).

Outro resultado interessante obtido tem relação ao total de ortólogos inferidos em relação à quantidade de proteínas de cada organismo, o que oferece uma visão sobre o percentual de proteínas que formam ortólogos com a base utilizada. A Tabela 4.2 detalha este resultado.

Tabela 4.2: Total de ortólogos inferidos através do elastic-OrthoSearch, por organismo, com a relação de percentual frente ao total de proteínas de seu respectivo proteoma.

| **Organismo** | **Proteínas** | **Ortólogos inferidos** | **Ortólogos / Proteínas** |
|---|---|---|---|
| *Cryptosporidium hominis* | 3.885 | 2.555 | 65,77% |
| *Entamoeba histolytica* | 7.973 | 2.959 | 37,11% |
| *Leishmania infantum* | 7.872 | 3.885 | 48,97% |

Até o presente momento, a máquina de *workflow* criada para a execução do elastic-OrthoSearch em nuvem foi publicada em seminário internacional (Carrión et al., 2014) e também está presente nesta tese na forma do Anexo 8.1. Por se tratar de espaço resumido para publicação à época, os detalhes de projeto e desenvolvimento da mesma foram submetidos à revista "*Journal of Systems and Software"* e no momento estão sob revisão. O manuscrito é apresentado nesta tese como o Anexo 8.4.

## 4.2 **Metodologia para a criação de bases de ortólogos aprimoradas**

O primeiro lote de experimentos foi realizado com as bases (i) KO, (ii) EggNOG KOG (ii) e (iii) ProtozoaDB, conforme Figura 4.3 para inferência de ortólogos contra os organismos *Cryptosporidium hominis*, *Entamoeba histolytica e Leishmania infantum*.

Nesta etapa, foi utilizada a versão aprimorada do OrthoSearch, produto desta tese, como base para comparação futura para com as novas bases de ortólogos criadas pela metodologia proposta nesta tese e com o OrthoMCLDB.

Dos três organismos confrontados, o melhor resultado ocorreu com o ProtozoaDB, que inferiu 2.830 grupos ortólogos em *Cryptosporidium hominis*, 4.952

para *Entamoeba histolytica* e 3.821 em *Leishmania infantum*. Os demais resultados estão disponíveis na Figura 4.3.

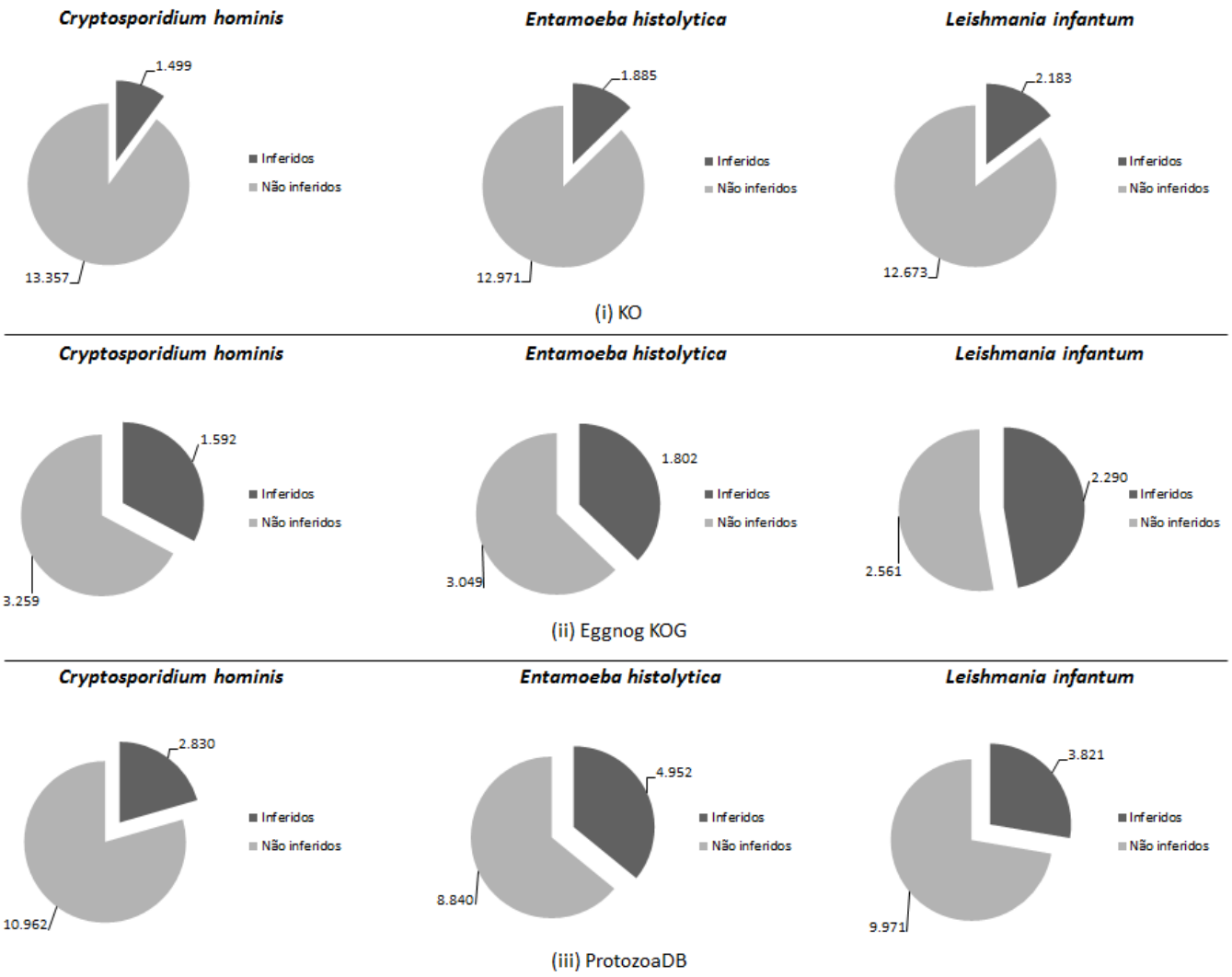


Figura 4.3: Ortólogos inferidos pela metodologia proposta, por base de dados e organismo em relação ao total de ortólogos disponíveis por base de dados - (i) KO, (ii) EggNOG KOG e (iii) ProtozoaDB.

Os resultados obtidos para cada espécie e cada base de ortólogos foram processados através de um *script* desenvolvido em linguagem R (R Project, 2015), com o objetivo de criar Diagramas de Venn que demonstrassem o quantitativo de grupos ortólogos inferidos pelo OrthoSearch que são espécie-específicos, compartilhados entre duas espécies e aqueles que são comuns aos três organismos e compõem o denominado núcleo, como demonstrado na Figura 4.4.

Dentre as três bases, EggNOG KOG apresenta o núcleo com o maior percentual de cobertura – a relação entre grupos ortólogos comuns às três espécies e o total de grupos ortólogos inferidos, com 26,72% de cobertura (828/3.099 grupos ortólogos). Estima-se que isso ocorra pois EggNOG KOG possui menor diversidade de táxons frente a KO, o que em teoria, poderia facilitar a formação de um *core* maior de ortólogos.

Em segundo lugar, KO apresentou 20,94% de cobertura, com 719 grupos no núcleo frente a total de 3.434 grupos inferidos, seguido por ProtozoaDB com 4,65% de cobertura (464/9.979 grupos), o que pode ser associado à mesma possuir menor diversidade de *táxons*, porém com maior especificidade dos mesmos (apenas protozoários), o que favoreceria a presença de genes espécie-específicos.

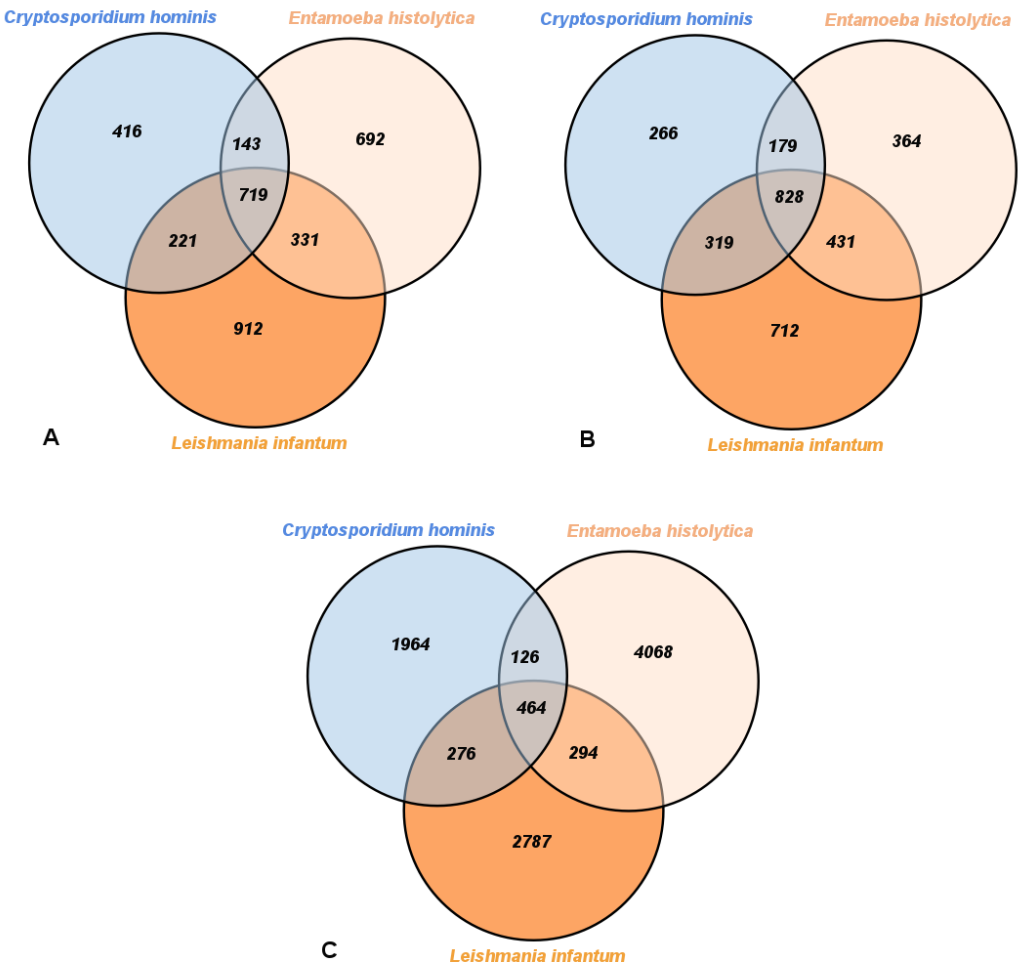


Figura 4.4: Grupos de genes espécie-específicos, ortólogos par-a-par e comuns aos três organismos, por base de ortólogos utilizada - (A) KO, (B) EggNOG KOG e (C) ProtozoaDB.

Os resultados mais expressivos em relação aos genes espécie-específicos foram todos obtidos através do ProtozoaDB, com 40,77% de cobertura para *Entamoeba histolytica* (4.086/9.979 grupos), seguida por *Leishmania infantum* com 27,93% (2.787/9,979 grupos) e por fim *Cryptosporidium hominis* com 19,68% de cobertura (1.964/9.979 grupos).

Com a aplicação da metodologia para criação de bases de ortólogos proposta nesta tese, foi possível criar duas novas bases, a saber: “KO + EggNOG KOG” e “KO + EggNOG KOG + ProtozoaDB”. Conforme apresentado na Tabela 4.3, cada base conta com grupos que permaneceram intactos e grupos da base original que foram aumentados a partir dos resultados obtidos contra uma segunda base de ortólogos.

Tabela 4.3: Detalhes relativos ao processo de criação e composição dos grupos ortólogos de cada base criada com a metodologia proposta.

| Etapa de criação de base | | Ortólogos na base original | Ortólogos na base confrontada | Ortólogos intactos na base original | Ortólogos intactos na base confrontada | Grupos aumentados | Ortólogos na nova base |
|---|---|---|---|---|---|---|---|
| Base original | Base confrontada | | | | | | |
| KO | EggNOG KOG | 14.856 | 4.851 | 2.087 | 2.082 | 2.769 | 16.938 |
| KO + EggNOG KOG | ProtozoaDB | 16.938 | 13.792 | 3.909 | 10.763 | 3.029 | 27.701 |

A criação de “KO + EggNOG KOG” caracteriza oferta de 14,02% grupos ortólogos a mais do que KO (16.938/14.856 grupos). Além disso, 18,63% dos grupos de KO foram expandidos, em total de 2.769 grupos.

Em sequência, quando “KO + EggNOG KOG” é confrontada com ProtozoaDB, a nova base criada, “KO + EggNOG KOG + ProtozoaDB” garantiu oferta 63,54% maior de grupos ortólogos frente a “KO + EggNOG KOG” (27.701/16.938), com expansão de 17,88% grupos de “KO + EggNOG KOG”.

Os detalhes sobre cada uma das bases criadas pela metodologia proposta podem ser observados na Tabela 4.4. A base “KO + EggNOG KOG + ProtozoaDB”, quando comparada a KO, fornece 86,46% mais grupos ortólogos (27.701/14.856) e 22,45% mais proteínas (2.582.631/2.109.027).

Tabela 4.4: Características de cada base de ortólogos criada com a metodologia proposta.

| Base de ortólogo / Característica | Total de grupos ortólogos | Total de proteínas | Volume de dados da base (GBytes) |
|---|---|---|---|
| KO + EggNOG KOG | 16.938 | 2.518.449 | 1,4 |
| KO + EggNOG KOG + ProtozoaDB | 27.701 | 2.582.631 | 1,5 |

Adicionalmente, a Tabela 4.5 detalha como os protozoários, organismos de interesse do LBCS/IOC estão representados em cada uma das bases de ortólogos utilizadas nestes experimentos de criação de novas bases.

Neste sentido, os resultados demonstram acréscimo de 61,98% de KO para “KO+EggNOG KOG”, partindo de 3.612 para 5.851 grupos que contém ao menos uma proteína de protozoário; até crescimento de 379% neste mesmo parâmetro,

quando comparadas as bases KO e “KO + EggNOG KOG + ProtozoaDB”, em salto de 3.612 para 17.305 grupos com contribuição de protozoários.

Além disto, o número total de proteínas de protozoários nestas duas bases criadas também foi expressivo, com um primeiro salto de 162% (de 46.027 para 74.630) de KO para “KO + EggNOG KOG” e em seguida ao máximo de 300% de KO para “KO + EggNOG KOG + ProtozoaDB”, de 46.027 para 138.814 proteínas de protozoários.

Tabela 4.5: Contribuição de proteínas de protozoários em cada base de ortólogos.

| Base de ortólogos | Total de grupos ortólogos | Grupos ortólogos com ao menos uma proteína de protozoário | Total de proteínas da base | Total de proteínas de protozoários |
|---|---|---|---|---|
| KO | 14.856 | 3.612 (24,31%) | 2.108.653 | 46.027 (2,18%) |
| KO + EggNOG KOG | 16.938 | 5.851 (34,54%) | 2.518.449 | 74.630 (2,96%) |
| KO + EggNOG KOG + ProtozoaDB | 27.701 | 17.305 (62,47%) | 2.582.631 | 138.814 (5,37%) |

Uma vez criadas as novas bases de ortólogos, foram repetidos os experimentos de inferência de ortólogos com os três protozoários supracitados e seus resultados analisados em comparação com a base de ortólogos de partida, KO.

A Figura 4.5 demonstra os resultados obtidos para cada uma das bases de ortólogos criada pela metodologia, para cada organismo utilizado.

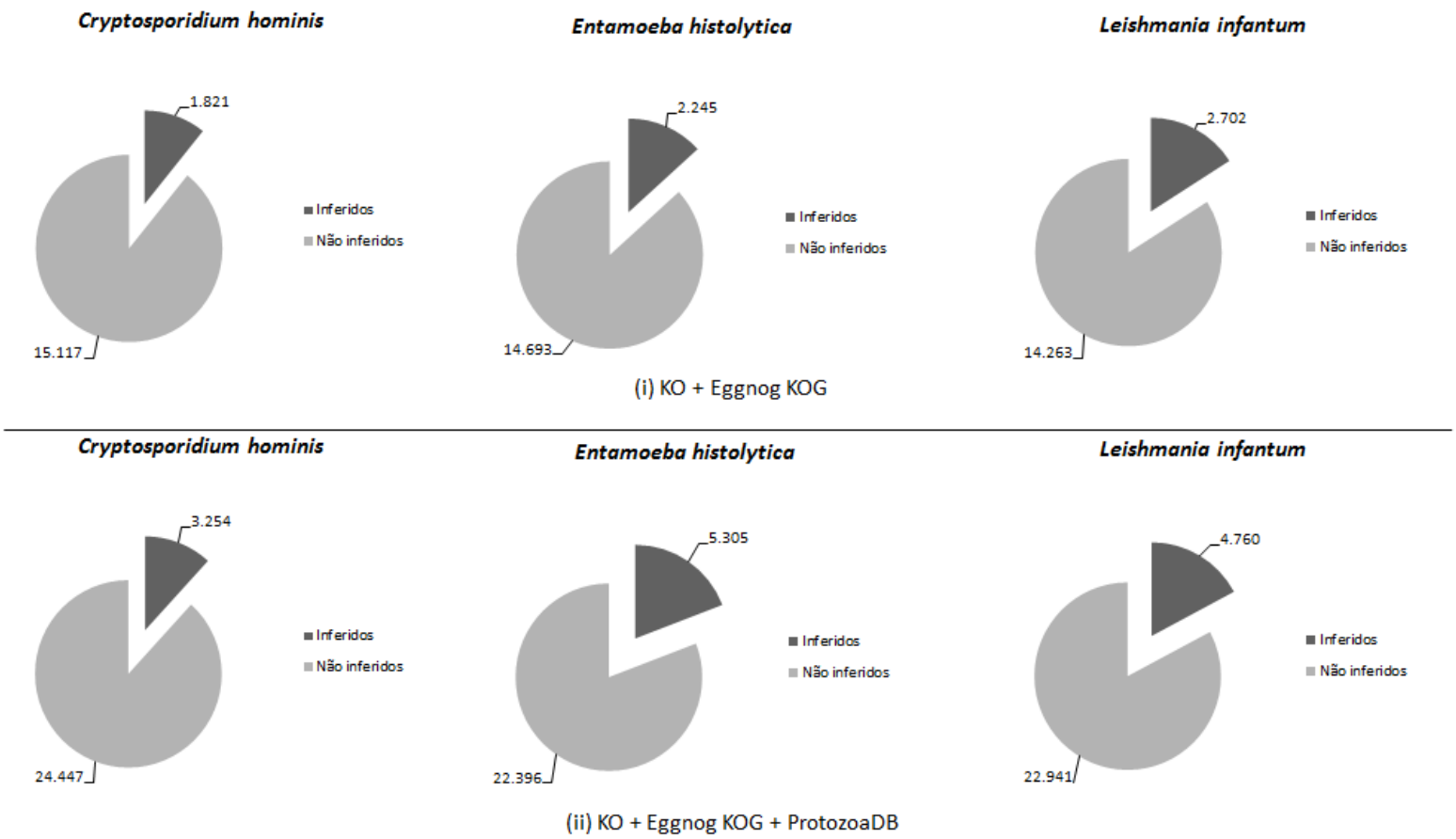


Figura 4.5: Ortólogos inferidos pela metodologia proposta, por base de dados e organismo em relação ao total de ortólogos disponíveis por base de dados - (i) “KO + EggNOG KOG” e (ii) “KO + EggNOG KOG + ProtozoaDB”.

Nossa metodologia possibilitou o aumento em 86,47% do total de grupos ortólogos ofertados e de 22,45% mais proteínas quando comparamos “KO + EggNOG KOG + ProtozoaDB” com KO.

Em todos os casos observa-se aumento na quantidade de grupos ortólogos inferidos, em todos os organismos, contra todas as bases utilizadas.

Os melhores *hits* recíprocos obtidos entre os organismos e as bases (A) “KO + EggNOG KOG” e (B) “KO + EggNOG KOG + ProtozoaDB” foram consolidados e representados em Diagramas de Venn, conforme a Figura 4.6.

Estima-se que o *core* obtido entre os três organismos tenha diminuído ao utilizar a maior base de dados criada com a metodologia (“KO + EggNOG KOG + ProtozoaDB”) devido à mesma possuir maior diversidade de *táxons* frente à base “KO + EggNOG KOG”.

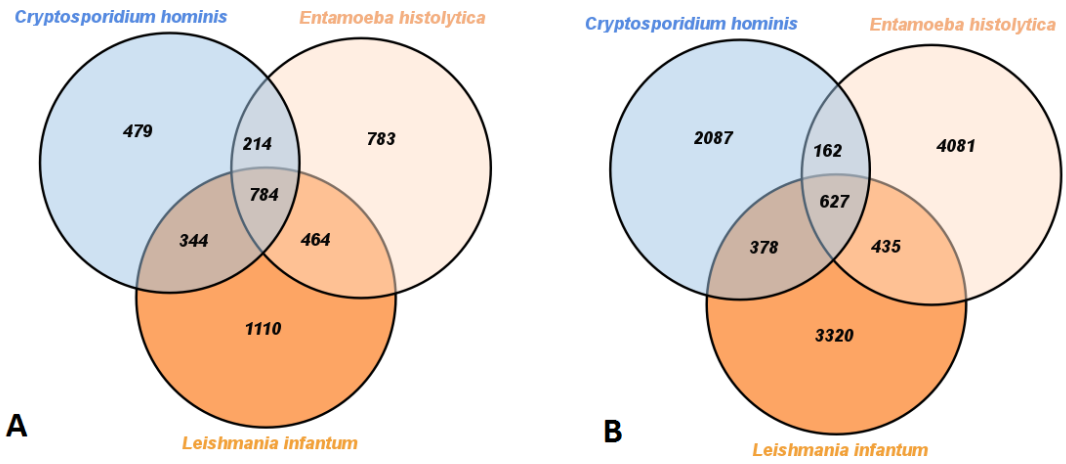


Figura 4.6: Genes espécie-específicos, compartilhados par-a-par e comuns aos três organismos, por base de ortólogos criadas com a metodologia proposta - (A) “KO + EggNOG KOG” e (B) “KO + EggNOG KOG + ProtozoaDB”.

Os resultados detalhados de quantidade de grupos ortólogos inferidos por base de dados utilizada (melhores *hits* recíprocos) estão disponíveis na Tabela 4.6. “KO + EggNOG KOG” apresentou o melhor resultado em relação ao total de grupos ortólogos comum aos três organismos frente ao total de *hits* obtidos – 11.089 ortólogos - com 18,76% (784/4.178 ortólogos), seguido por “KO + EggNOG KOG + ProtozoaDB”, com 5,65% (627/11.089 ortólogos).

O melhor resultado espécie-específico ocorreu com *Entamoeba histolytica* e “KO + EggNOG KOG + ProtozoaDB”, com 36,80% de *hits* (4.081/11.089), seguida por *Leishmania infantum*, com 29,94% (3.320/11.089) e *Cryptosporidium hominis*, com 18,81% (2.086/10.089), ambos com a mesma base.

Tabela 4.6: Detalhamento da inferência de ortólogos com visão de genes espécie-específicos, par-a-par e comum aos três protozoários, por base de ortólogos.

| | *Cryptosporidium hominis* | | *Entamoeba histolytica* | | *Leishmania infantum* | | *Cryptosporidium hominis and Entamoeba histolytica* | | *Cryptosporidium hominis* and *Leishmania infantum* | | *Entamoeba histolytica* and *Leishmania infantum* | | Core | | Total | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| KO | 416 | 12,11% | 692 | 20,15% | 912 | 26,56% | 143 | 4,16% | 221 | 6,44% | 331 | 9,64% | 719 | 20,94% | 3,434 | 100,00% |
| KO + EggNOG KOG | 479 | 11,46% | 783 | 18,74% | 1,110 | 26,57% | 214 | 5,12% | 344 | 8,23% | 464 | 11,11% | 784 | 18,76% | 4,178 | 100,00% |
| KO + EggNOG KOG + ProtozoaDB | 2,086 | 18,81% | 4,081 | 36,80% | 3,320 | 29,94% | 162 | 1,46% | 378 | 3,41% | 435 | 3,92% | 627 | 5,65% | 11,089 | 100,00% |

Para efeito comparativo entre os resultados obtidos pela nossa metodologia e base criadas, foram inferidos os totais de grupos ortólogos para cada um dos organismos utilizados nos experimentos desta tese, agora através de consultas filéticas diretamente na página do OrthoMCLDB (OrthoMCL, 2015).

Estas consultas informam que no OrthoMCLDB existem 3.516 grupos ortólogos que contém ao menos uma proteína de *Cryptosporidium hominis*, 6.107 para *Entamoeba histolytica* e 7.538 para *Leishmania infantum*.

Em seguida, foi elaborada consulta para identificação de interseção entre os grupos e organismos supracitados e montado Diagrama de Venn para representação gráfica dos resultados obtidos, conforme a Figura 4.7.

O núcleo comum aos três organismos é composto de 760 grupos ortólogos, o que representa 5,12% do total de ortólogos inferidos (760/14.846). Os grupos espécie-específicos para *Cryptosporidium hominis* correspondem a 15,76% do total (2.340/14.846); em *Entamoeba histolytica*, 32,44% do total (4.816/14.846); para *Leishmania infantum*, 41,39% (6.145/14.846).

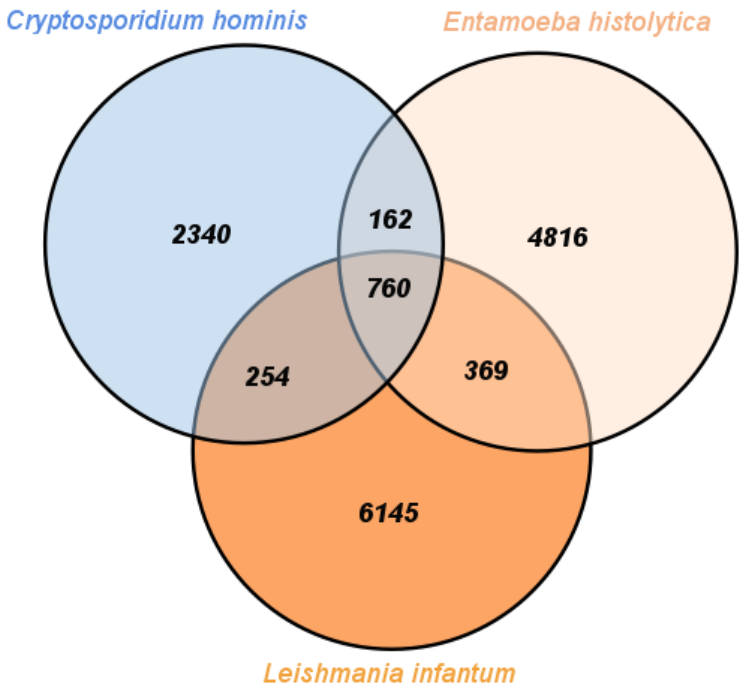


Figura 4.7: Genes espécie-específicos, compartilhados par-a-par e comuns aos três organismos inferidos com o OrthoMCLDB.

Para a identificação de alvos potenciais de fármacos para *Leishmania* spp., foi realizada consulta via BlastP contra a maior base de ortólogos criada através de nossa metodologia, “KO + EggNOG KOG + ProtozoaDB”. Foram inferidos 7.622

grupos ortólogos (27,5% do total – 27.701 grupos) que não obtiveram *hit* contra o proteoma humano.

Dentre os mesmos, 1.805 grupos (6,5%) pertencem ou a KO ou a EggNOG KOG, mas não estão disponíveis no ProtozoaDB, que contém apenas proteínas de protozoários, dentre os quais, *Leishmania* spp..

Destes, 13 grupos ortólogos, o que corresponde 0,05% do total, contém ao menos uma proteína de *Leishmania* spp., conforme apresentado na Tabela 4.7. Estes grupos ortólogos podem ser considerados como alvos potenciais de fármacos em *Leishmania* spp..

Tabela 4.7: Grupos ortólogos que podem contribuir com alvos potenciais em *Leishmania* spp.

| **Grupo ortólogo** | **Anotação** |
|---|---|
| K14118.cdhit | *energy-converting hydrogenase B subunit I* |
| K13666.cdhit | *UDP-GlcNAc:polypeptide alpha-N-acetylglucosaminyltransferase* |
| K03668.cdhit | *heat shock protein HslJ* |
| K08907.cdhit | *light-harvesting complex I chlorophyll a/b binding protein 1* |
| K13690.cdhit | *alpha-1,3-mannosyltransferase* |
| K13674.cdhit | *phosphoglycan alpha-1,2-arabinopyranosyltransferase* |
| K08276.cdhit | *ecotin* |
| K02817.cdhit | *PTS system, trehalose-specific IIB component* |
| K01833.cdhit | *trypanothione synthetase/amidase* |
| K03329.cdhit | *hypothetical protein* |
| K01973.cdhit | *mitochondrial RNA editing ligase 1* |
| K03356.cdhit | *anaphase-promoting complex subunit 9* |
| K13672.cdhit | *galactofuranosyltransferase* |

A mesma consulta foi realizada através do BlastP, contra cada uma das bases de ortólogos em suas composições originais. Os resultados são listados na Tabela 4.8, que aponta grupos ortólogos que não possuem similaridade com o proteoma humano e que contém, minimamente, uma sequência de *Leishmania* spp.

Tabela 4.8: Total de grupos ortólogos com sequências de *Leishmania* spp. em cada base de ortólogos, em sua composição original, não presentes em humanos.

| **Base de ortólogos** | **Grupos ortólogos** |
|---|---|
| KO | 5 |
| Eggnog KOG | 1 |
| ProtozoaDB | 1.524 |

# 5 DISCUSSÃO

Esta tese apresenta duas contribuições principais: uma proposta de solução de genômica comparativa para ambiente computacional distribuído – elastic-OrthoSearch; e a proposta de uma nova metodologia para criação de bases de ortólogos aprimoradas.

Ambas tiveram origem no OrthoSearch, desenvolvido pelo LBCS/IOC/Fiocruz, para a inferência de ortólogos, sendo a primeira o resultado de colaboração entre o mesmo e o I3M/UPV, como produto do período de doutorado sanduíche no exterior do autor desta tese.

Atualmente existem distintas opções para fornecer extensa capacidade de processamento e armazenamento computacional para a realização de experimentos. Dentre elas, temos o uso de *clusters*, *grids* e a nuvem.

A nuvem atua como uma entidade abstrata, que pode prover recursos computacionais ao solicitante na configuração e quantidade desejadas, sem a necessidade de hospedagem física, configuração e manutenção dos mesmos pelo solicitante (Armbrust et al., 2010; Mell & Grance, 2011).

A escolha pelo ambiente de nuvem computacional segue a lógica de utilização de recursos e infraestrutura de forma consciente e sob demanda, da otimização de investimentos em pesquisa e desenvolvimento e da economicidade de uma plataforma computacional.

Em contrapartida ao investimento em infraestrutura de tecnologia da informação – servidores, armazenamento, equipamentos de rede – e de recursos humanos para a instalação, configuração e gestão dos mesmos, a nuvem oferece a oportunidade de solicitar e trabalhar com recursos apenas durante o momento de execução dos experimentos.

Segundo Stein (Stein, 2010), existem alguns pontos críticos a favor da nuvem: a gestão do extenso volume de dados gerados pelas tecnologias de NGS, a complexidade associada à manutenção de infraestrutura computacional própria e a necessidade de recursos para experimentos que ocorrem sob demanda.

Com o grande volume de dados gerados (superiores a ordem de *Terabytes*), mesmo em redes com alto desempenho e elevada largura de banda, a movimentação de dados entre servidores, *clusters*, *grids* ou estações de trabalho é penosa para a execução de experimentos.

Além disso, a gestão destes ambientes é complexa e custosa e deve levar em conta a alocação (e por muitas vezes a replicação) destes dados de forma local. Ainda, o gerenciamento dos níveis de acesso aos mesmos e a realização de cópias de segurança (*backups*), entre outras atividades, são outros tópicos que demandam mais investimento e manutenção em comparação à nuvem.

Outras ineficiências citadas por Stein (Stein, 2010) residem na necessidade de manutenção e atualização de um ambiente computacional próprio para atender a execução de experimentos e no tempo ocioso de tal cenário, uma vez que os experimentos ocorrem sob demanda. Infraestrutura computacional, mesmo que ociosa, requer manutenção, monitoramento e consome recursos como eletricidade, por exemplo.

Shanker (Shanker, 2012) reforça estes pontos e destaca, a favor da nuvem, a capacidade de virtualização de recursos computacionais, que permite criar e disponibilizar instâncias (nós computacionais) que podem ser dinamicamente configurados, entregues e depois liberados para experimentos, conforme a demanda.

Outra plataforma tradicionalmente voltada para a execução de experimentos em larga escala, o *grid*, é comparada com a nuvem por Foster em (Foster et al., 2008). De forma simplificada, os *grids* surgiram como uma forma de oferta de recursos computacionais em larga escala para organizações, usualmente concebidos na forma de infraestruturas computacionais privadas.

Para sua utilização, é necessária a adoção de um sistema gerenciador de filas de trabalho, para o qual os usuários devem solicitar recursos durante determinado período de tempo, este informado *a priori* a execução de seus experimentos, o que não é necessário na nuvem.

Finalmente, a abstração fornecida por uma nuvem ao usuário é de extrema relevância, pois minimiza preocupações para com o conhecimento e gestão de recursos computacionais, redes e armazenamento, entre outros aspectos que não compõem o núcleo de atividades fim de diversos experimentos.

Portanto, a abstração fornecida e o baixo custo associado à utilização de recursos virtuais sob demanda – nuvem – são alguns dos grandes benefícios frente à adoção de infraestruturas como o *cluster* ou *grids*, que requerem investimento e manutenção permanentes, correm o risco de tornarem-se obsoletos com o tempo e podem terminar por permanecer em estado de ociosidade conforme inexistam experimentos a todo o momento.

A construção de *workflows* científicos, tais como o elastic-OrthoSearch, pode ser suportada, por distintos SWfMS, tais como Galaxy (Goecks et al., 2010), Taverna (Oinn et al., 2004), Kepler (Altintas et al., 2004; Ludäscher et al., 2006), entre outros.

Usualmente, soluções dessa natureza prezam por serem conectáveis a uma nuvem comercial, como a Amazon (Amazon, 2012) e oferecerem nós virtuais para processamento, mas não possuem foco na adaptação de uma aplicação para execução de forma distribuída e/ou paralela.

Até o momento, para o problema estudado, tais SWfMS existentes ora não oferecem suporte para execução na nuvem, ora o fazem de forma precária ou específica, como no caso do próprio Galaxy e da iniciativa Tavaxy (Abouelhoda et al., 2012).

Nos últimos anos, nota-se o crescente interesse em iniciativas voltadas para a criação de soluções de genômica comparativa em nuvem computacional, tais como Crossbow (Gurtowski et al., 2012; Langmead et al., 2009a), CloudBlast (Matsunaga et al., 2008) e RSD (Wall et al., 2010; Wall & Deluca, 2007). Muitas destas ferramentas oferecem suporte à execução distribuída através da metodologia MapReduce (Dean & Ghemawat, 2008; Taylor, 2010).

Entende-se que seria possível construir uma aplicação MapReduce do zero, contudo, por esta tese ter caráter colaborativo entre duas instituições acadêmicas (LBCS/IOC/Fiocruz e I3M/UPV), houve espaço para apresentação de propostas para o desenho da solução entre as partes envolvidas para adequar as expectativas às habilidades e competências de todos os envolvidos.

Por tais razões, foi preterida a construção de uma aplicação MapReduce, tão como a utilização de um SWfMS pré-existente, em preferência ao desenvolvimento de uma máquina de *workflow* própria (Carrión et al., 2014).

Esta máquina de *workflow* que suporta o elastic-OrthoSearch foi projetada com base em seis eixos (alguns ainda em desenvolvimento), descritos no Anexo 8.4, que resumidamente fornecem: (i) independência de plataforma; (ii) facilidade de uso; (iii) suporte a distintos tipos de *workflow* DAG; (iv) estágios modulares independentes; (v) suporte a plataformas computacionais distintas; e (vi) aderência à nuvem.

A máquina de *workflow* requer um arquivo de configuração escrito na linguagem JSON (ECMA, 2013) para a especificação dos estágios do *workflow*. Este formato foi escolhido ao invés da linguagem XML (XML, 2008) por não utilizar marcadores de fim de seção (*end tags*), ser intuitivo, curto e de escrita e leitura

simples.

Atualmente este arquivo JSON precisa ser fornecido diretamente pelo usuário para a execução do elastic-OrthoSearch devido à ausência de interface gráfica para a máquina de *workflow*.

Com relação ao armazenamento dos dados para o elastic-OrthoSearch, foi utilizado armazenamento em rede, na estrutura de NFS, o que tornou os dados disponíveis para todas as instâncias que os necessitassem.

Os experimentos para a validação do elastic-OrthoSearch foram realizados através da utilização de três bases de ortólogos com características e composição distintas: KO (Kanehisa et al., 2012; Kegg Orthology, 2012), EggNOG euNOG (Powell et al., 2013) e ProtozoaDB (Dávila et al., 2008). Os grupos da base KO têm representatividade em toda a árvore da vida, enquanto EggNOG euNOG contém apenas grupos compostos de proteínas de eucariotos e ProtozoaDB apenas de protozoários.

Estas foram confrontadas contra três protozoários selecionados de forma aleatória: *Cryptosporidium hominis*, *Entamoeba histolytica* e *Leishmania infantum*. Os dados de proteínas desses organismos foram obtidos através do ProtozoaDB (Dávila et al., 2008).

A máquina de *workflow* desenvolvida em parceria com o I3M/UPV (Carrión et al., 2014) registrou o tempo decorrido para a execução de cada estágio do elastic-OrthoSearch, incluídas as etapas de criação, alocação e liberação de máquinas virtuais e a transferência de dados para cada estágio, o que também pode ser útil em atividades futuras para registro de dados de proveniência.

Especificamente, a aceleração mínima obtida com o uso da nuvem foi de 1,99 vezes o tempo de execução fora da nuvem, no cenário com duas instâncias contra *Leishmania infantum*. O melhor resultado obtido chegou a ser 4,99 vezes mais rápido, para 16 instâncias contra *Cryptosporidium hominis*. Estes resultados são encorajadores para a realização de experimentos com mais organismos e bases de dados.

Os resultados obtidos através do elastic-OrthoSearch sinalizam para um cenário associado aos conceitos descritos na Lei de Amdahl (Amdahl, 1967) para sistemas distribuídos/paralelos, que cita limites para a aceleração de determinada aplicação, referente às etapas não paralelizáveis da mesma.

Existem indícios de que os algoritmos sequenciais envolvidos no elastic-OrthoSearch (por exemplo, hmmpress e hmmscan) impõem um teto para a escalabilidade e consequente aceleração máxima possível. Esta ainda não foi alcançada, mas já demonstra sinais de sua ocorrência em cenários que envolvam quantidade adicional de instâncias computacionais.

Conforme foi possível observar, ainda de forma empírica, nos experimentos realizados na nuvem, estima-se, pelo padrão da curva obtida, que existirá um número ideal de instâncias (nós computacionais) que representarão ganho no tempo de execução do *workflow*. Especificamente, estima-se que a partir de 16 instâncias computacionais, o ganho obtido não seguiria o padrão observado até então.

Além disso, a dependência de tarefas também é fator de atenção, como no caso de alinhamentos de grupos ortólogos extensos pelo Mafft, que devem ser realizados previamente à construção de perfis HMM e assim podem penalizar o tempo total de execução do *workflow*.

Mesmo com as limitações associadas à capacidade alcançar um número ideal de instâncias, o desempenho obtido com o elastic-OrthoSearch é expressivo e os custos para prover instâncias em uma nuvem tendem a ser menores do que o investimento necessário para a aquisição, configuração e manutenção de *clusters* ou servidores de grande porte.

Ademais, as tarefas que compõem o elastic-OrthoSearch e que ainda são sequenciais podem ser alvos de pesquisa e modificação para que adotem técnicas de programação distribuída/paralela no futuro.

O elastic-OrthoSearch tem o potencial de alavancar experimentos para inferência de ortólogos, uma vez que foi possível obter ganhos expressivos com sua execução na nuvem.

A diversidade e a complexidade dos problemas e questões a serem respondidas pela genômica comparativa, aliadas ao amadurecimento de tal ambiente computacional (Armbrust et al., 2010; Mell & Grance, 2011) e à crescente disponibilidade de ferramentas de programação em ambiente distribuído (Apache, 2013; Gabriel et al., 2004; MPI, 2013; Taylor, 2010; The MPI Forum, 1993) justificam o investimento em ações de pesquisa e desenvolvimento para soluções como o elastic-OrthoSearch.

Sugere-se que futuramente estas etapas sejam alvo de pesquisa, ora através de alterações diretas nos algoritmos ou na adoção de aplicativos que realizem a mesma tarefa, mas que sejam distribuídos/paralelos.

Adicionalmente, a solução proposta não obriga o cientista a investir na aquisição, configuração e manutenção de infraestrutura computacional para a realização de experimentos.

Existem aspectos ainda não explorados, como a questão de segurança, ainda não abordada devido à heterogeneidade de ambientes possíveis para a realização dos experimentos.

Acredita-se que, com um maior volume de experimentos realizados no futuro, maiores serão as contribuições fornecidas pelos usuários para amadurecer a solução.

Até o momento a máquina de *workflow* que suporta o elastic-OrthoSearch não oferece recursos de proveniência de dados, mas nos permite rastrear em que estágio da execução o experimento se encontra, e se já concluída, quanto tempo decorreu. Estas informações são armazenadas em um banco de dados MongoDB (MongoDB, 2015) e existe a intenção em adotar mecanismos que permitam realizar pausas e retomadas de execução de experimentos.

Finalmente, a solução elastic-OrthoSearch é um trabalho em contínuo processo de aprimoramento. Até o momento já foi possível executá-la sob uma nuvem computacional e um *cluster*, mas ela pode ser estendida para utilização em outros ambientes, como *grids*, se assim desejado.

No período inicial de desenvolvimento desta tese, foi necessário conhecer o OrthoSearch em sua versão original e entender quais seriam as possibilidades para adaptá-lo a um ambiente computacional novo, a nuvem.

Uma alternativa para tal foi identificar como torná-lo independente de soluções do tipo SWfMS e como utilizar linguagens de programação que permitissem potencializar seu desempenho, mesmo que ainda fora da nuvem e que pudessem estar mais próximas ao cenário do LBCS/IOC/Fiocruz à época.

Assim, o OrthoSearch foi atualizado e foram adotadas as linguagens de programação C++ e Ruby. Além disso, foi possível adotar a versão 3 do pacote HMMER.

No primeiro conjunto de experimentos, ainda prévios à solução elastic-OrthoSearch e à metodologia para a futura criação de novas bases de ortólogos, foram confrontados três protozoários (*Cryptosporidium hominis*, *Entamoeba*

*histolytica* e *Leishmania infantum*) perante três bases de ortólogos (KO, EggNOG KOG e ProtozoaDB).

Neste momento, ressalta-se que a base EggNOG ainda estava em sua versão 3.0, (diferentemente da versão 4.0, lançada posteriormente e utilizada nos testes do elastic-OrthoSearch) e os grupos ortólogos de eucariotos eram denominados KOG, enquanto na versão 4.0 passaram a ser declarados como euNOG.

Os primeiros resultados obtidos foram analisados sobre o ponto de vista quantitativo (*hits* obtidos contra cada organismo) e sobre diagramas de Venn elaborados com o auxílio da linguagem R (R Project, 2015), que permitiram uma visualização de *hits* espécie-específicos, compartilhados por dois organismos, e o núcleo, comum a todos os organismos estudados.

Estas bases possuem características próprias, o que pode ter influenciado os resultados obtidos. Por exemplo, enquanto os grupos da base KO têm representatividade em toda a árvore da vida, o subconjunto EggNOG KOG contém apenas grupos compostos de proteínas de eucariotos e ProtozoaDB apenas de protozoários.

Neste sentido, o *core* – ortólogos comuns aos três organismos - obtido na execução do OrthoSearch com a base KO pode ser considerado pequeno (719 *hits*) quando comparado a quantidade total de grupos ortólogos da mesma (14.856 grupos) e a quantidade de melhores *hits* recíprocos obtidos (3.434 *hits*). Como esta base contempla organismos de toda a árvore da vida, a distância evolutiva entre os mesmos poderia elevar a dificuldade em inferir grupos ortólogos.

Por outro lado, quando a base EggNOG KOG é confrontada, apresenta um *core* maior em relação ao total de *hits* inferidos (829/3.099), o que pode ser creditado por esta ser composta apenas de organismos eucariotos.

Por fim, a base de dados ProtozoaDB fornece excelente quantitativo de melhores *hits* espécie-específicos dentre esses três primeiros experimentos e o menor *core* dentre os três (464/9.979). Estima-se que isso tenha ocorrido devido à composição da base ProtozoaDB – apenas organismos protozoários – e ao fato dos organismos confrontados também serem protozoários. Sendo assim, as chances de se obter um *hit* espécie-específico seriam mais elevadas.

Comparativamente, a relação entre o total de ortólogos inferidos e o total de grupos disponíveis para inferência de cada base chegou a 47% para *Leishmania*

*infantum* contra EggNOG KOG (2.290/4.851) e de no mínimo aproximadamente 10% para *Cryptosporidium hominis* contra KO (1.499/14.856).

Os bons resultados obtidos com o uso do OrthoSearch já foram demonstrados em outro estudo, realizado por Cuadrat e colaboradores (Cuadrat et al., 2014), que confrontaram cinco protozoários contra as bases de dados COG/KOG do NCBI (Tatusov et al., 2003)

Durante a realização de tais experimentos com o OrthoSearch e com o foco nas possibilidades de criação do que futuramente seria o elastic-OrthoSearch, foi alcançada uma contribuição biológica relevante e não inicialmente planejada como objetivo principal, uma metodologia para criação de novas bases de grupos ortólogos, publicada em revista internacional (Kotowski et al., 2015).

Para tal, ao invés de confrontar uma base de ortólogos e o proteoma de um organismo – um arquivo com todas as suas proteínas em formato *multifasta* – teve-se a ideia em confrontar diretamente duas bases de ortólogos, que são conjuntos de arquivos em formato *multifasta*.

Essa ideia surgiu após o entendimento de que, uma vez que o OrthoSearch entrega como resultado final uma lista com melhores *hits* recíprocos entre seus dados de entrada (base de ortólogos e organismo), se os mesmos passassem a ser duas bases de ortólogos, seria possível complementar uma base através de seu confronto para com outra.

Seja pela adição de novos grupos ortólogos que não obtiveram *hits* e poderiam então compor novos grupos na base primária, ou através da expansão de seus grupos originais, com o uso das proteínas elencadas como melhores *hits* recíprocos, pertencentes a grupos ortólogos da base confrontada, seria possível oferecer mais grupos, com maior variabilidade de proteínas.

Estas seriam compiladas em uma nova base de ortólogos, que poderia então ser utilizada em uma execução do OrthoSearch ou do próprio elastic-OrthoSearch, ou simplesmente ser disponibilizada para demais experimentos de inferência de homologia.

Os experimentos para a criação de novas bases de ortólogos utilizaram três bases: KO, EggNOG KOG e ProtozoaDB. Foram realizadas duas iterações com a metodologia proposta, que permitiram a criação de duas bases de ortólogos – “KO + EggNOG KOG” e “KO + EggNOG KOG + ProtozoaDB” - que contém grupos de ortólogos intactos provenientes das duas bases confrontadas ou grupos que foram

expandidos através dos resultados de inferência de ortologia providos pelo OrthoSearch.

Os grupos intactos provenientes da base confrontada contribuem diretamente na maior oferta de grupos ortólogos para a realização de experimentos, enquanto que os grupos expandidos têm o potencial de oferta de maior variabilidade de proteínas e organismos se comparados à sua composição original.

Além de prover meios para melhorar a inferência de ortólogos com o ProtozoaDB, este estudo iniciou a construção de novas bases através de KO e EggNOG KOG devido à variabilidade de organismos presente nas mesmas e o extenso volume de dados disponível para os experimentos.

Os identificadores de cada grupo ortólogo em sua base de dados, assim como a anotação funcional de cada proteína foram mantidos em suas formas originais, para auxiliar em futuras iniciativas de rastreabilidade/proveniência dos dados.

Esta proposta de metodologia foi capaz de criar novas bases em uma abordagem não intrusiva ao OrthoSearch, sem a obrigatoriedade de mudanças em suas funções e características próprias. Foram tratados apenas os dados de entrada e saída do mesmo.

A abordagem do OrthoSearch para inferência de ortólogos através de melhores *hits* recíprocos com o uso de perfis HMM, poderia em teoria, ter a capacidade de identificar mais homólogos distantes do que metodologias baseadas em BLAST, tais como o OrthoMCLDB.

Devido a esta possibilidade, foram identificados os quantitativos de homólogos inferidos pelo OrthoMCLDB contra os mesmos protozoários utilizados no OrthoSearch com a nova base “KO + EggNOG KOG + ProtozoaDB”, sob a ótica dos ortólogos espécie-específicos, par a par e comum a todos e tais resultados foram analisados para as duas metodologias.

Nesta análise, identificou-se que o OrthoMCLDB foi capaz de inferir maior quantitativo de ortólogos espécie-específicos para todos os três organismos, com uma variação de 12% (*Cryptosporidium hominis* – 2.340/2.087) a 85% (*Leishmania infantum* - 6.145/3.320) quando comparado aos resultados com o OrthoSearch.

Como o OrthoMCLDB contém muitos grupos com apenas parálogos internos de um mesmo organismo em relação ao total de grupos (20.853/124.740) (Chen et al., 2006) - e os três organismos estão na referida base – é possível que esta

característica tenha influenciado positivamente na inferência de ortólogos a favor do OrthoMCLDB.

No tocante aos ortólogos comuns a dois organismos, a metodologia proposta é capaz de igualar os resultados do OrthoMCLDB, com 162 ortólogos entre *Cryptosporidium hominis* e *Entamoeba histolytica*, ou de superar seus resultados, com 17,88% mais ortólogos para *Entamoeba histolytica* (435/369) até 48,81% para *Cryptosporidium hominis* e *Leishmania infantum*.

O OrthoMCLDB identificou número maior de ortólogos comuns aos três organismos frente à nossa base, com 760 contra 627 grupos. Isto pode ter ocorrido devido à quantidade total de ortólogos oferecida por cada base de dados: enquanto OrthoMCLDB contém 124.740 grupos, “KO + EggNOG KOG + ProtozoaDB” contém 27.701 grupos.

Porém, ao olhar a relação entre o total de ortólogos inferidos e a quantidade total de grupos disponível em cada base, o cenário muda a favor da metodologia proposta nesta tese, mesmo ao considerar o grupo restrito de organismos utilizados nos experimentos. Enquanto a proporção para o OrthoMCLDB é de apenas 0,06% (760/124.470), “KO + EggNOG KOG + ProtozoaDB” alcança 2,26% do total (627/27.701), um resultado 375% melhor que o obtido com OrthoMCLDB.

Estima-se que este resultado a favor da base criada com a metodologia proposta nesta tese esteja relacionada à sensibilidade da mesma (uso de perfis HMM) em relação ao OrthoMCLDB, que utiliza Blast, conforme citado por DeLuca e colaboradores em experimento similar (DeLuca et al., 2012).

Iniciativas como a metodologia proposta podem auxiliar na construção de uma base de dados que atue como um conjunto de referência ou *benchmark* para a inferência de ortólogos, o que é alvo de pesquisa em aberto (Chen et al., 2007).

Além disso, estudos de filogenômica que buscam a otimização de árvores (McCormack et al., 2012) e até para revisão da definição de espécies (Konstantinidis & Tiedje, 2005) podem beneficiar-se da maior oferta de grupos ortólogos das bases de ortólogos aprimoradas oferecidas pela metodologia proposta nesta tese.

A metodologia proposta oferece meios para a criação de bases de ortólogos através de uma abordagem com base em perfis HMM. Com isso, espera-se prover um conjunto expandido de proteínas homólogas distantes, que podem ser utilizadas para elevar as chances de inferir conhecimento a respeito dos organismos a serem confrontados.

Os experimentos realizados com as três espécies de protozoários supracitadas também trouxeram como possibilidade a inferência de alvos potenciais em protozoários e a melhor compreensão sobre a biologia dos mesmos. Por exemplo, o *core* obtido entre os três organismos pode permitir a avaliação de quais destas proteínas estão relacionadas ao *housekeeping* dos mesmos.

Os resultados obtidos com o BlastP (Camacho et al., 2009) permitiram a identificação de 13 possíveis alvos (ortólogos) em *Leishmania* spp. que não obtiveram *hit* contra o proteoma humano (Tabela 4 do Anexo 8.4). Algumas destas já são descritas na literatura como possíveis alvos de drogas, como por exemplo: tripanotiona (*trypanothione*) (Colotti et al., 2013) (K01833.cdhit), relacionada a mecanismos de defesa contra estresse oxidativo (Krauth-Siegel et al., 2003); e alfa-1,3-manosiltransferase (*alpha-1,3-mannosyltransferase*) (Garami & Ilg, 2001; Shinde et al., 2014) (K13690.cdhit), uma enzima essencial para adicionar manose em glicosilfosfatidil, relacionada a resistência contra miltefosina.

Outras proteínas foram identificadas como alvos e podem ser estudadas, tais como: a subunidade I da hidrogenase B (*energy-converting hydrogenase B subunit I*) (Tersteegen & Hedderich, 1999) (K14118.cdhit), encontrada em uma Archaea (*Methanothermobacter thermautotrophicus*), pertencente a um domínio relacionado a subunidade proteica MnhB da proteína de transmembrana do tipo antiporte de $Na^+/H^+$ (*MnhB subunit of Na+/H+ antiporter*), predito como uma proteína de membrana (Hiramatsu et al., 1998; Ito et al., 2001); e a galactofuranose transferase (*galactofuranosyltransferase*) (Huang & Turco, 1993) (K13672.cdhit), relacionada ao gene LPG1, que atua como um ligante para a adesão de macrófagos (Spath et al., 2000; Zhang et al., 2004).

A metodologia proposta foi capaz de alocar proteínas novas (que também podem ser evolutivamente distantes) aos grupos ortólogos das bases em suas composições originais, através da identificação de relações de ortologia que ainda não haviam sido descritas, como no caso da subunidade I da hidrogenase convertedora de energia (*energy-converting hydrogenase B subunit I*).

Mesmo em cenários preliminares, foi possível avaliar a metodologia apresentada para, por exemplo, atuar na identificação de alvos de proteínas em protozoários. Foram identificados 13 alvos e pretende-se estender os experimentos dos três organismos utilizados para todos os 22 protozoários atuais que constam no ProtozoaDB (BiowebDB, 2012; Dávila et al., 2008), o que representará um ganho de escala com potencial expressivo na obtenção de resultados.

Finalmente, juntas, as duas soluções propostas nesta tese apresentam relevantes contribuições biológicas, em caráter inédito por sua criação e desenvolvimento.

Através do elastic-OrthoSearch, foi possível oferecer uma solução de genômica comparativa que executa em um ambiente de vanguarda para a computação – a nuvem – que recebe atenção da academia com a necessidade de resolução de problemas complexos e diversos.

A metodologia proposta, que permitiu a criação de bases de ortólogos aprimoradas, carrega o potencial de ofertar maior quantidade de grupos e maior variabilidade de proteínas para futuros experimentos de inferência de homologia, como inclusive através do elastic-OrthoSearch.

Sendo assim, experimentos futuros que combinem o potencial das duas soluções - o elastic-OrthoSearch e as bases de ortólogos aprimoradas - poderão contribuir, em larga escala, na anotação de proteínas ainda não caracterizadas, na inferência de alvos de drogas, em protozoários ou demais organismos que impactam na saúde humana e inclusive na criação de outras bases de ortólogos.

# 6 CONCLUSÕES

Esta tese teve como objetivo principal propor uma solução para inferência de ortólogos em ambiente computacional distribuído. Foram realizados estudos dentro do contexto da genômica comparativa em tal ambiente e também estudos paralelos, que auxiliaram na elaboração de soluções que, juntas, compuseram os objetivos específicos desta tese.

Especificamente, como resultado desta tese, foi possível:

1. Estabelecer uma infraestrutura de nuvem computacional privada no LBCS IOC/Fiocruz, que pode ser utilizada tanto para as soluções aqui apresentadas, quanto para futuras demandas de usuários.
2. Desenvolver a solução elastic-OrthoSearch, um *workflow* científico de genômica comparativa para inferência de ortólogos que executa sobre a nuvem, um ambiente computacional distribuído.
3. Construir novas bases de dados de grupos ortólogos através de uma nova metodologia com base no OrthoSearch. Para chegar a esta metodologia, o OrthoSearch foi atualizado para a adoção do pacote HMMER3 e com a independência de sistemas gerenciadores de *workflows*. Estas bases podem ser utilizadas para futuros experimentos de inferência de homologia entre organismos, além da própria criação de bases adicionais.

Além de alcançar o objetivo de criar uma solução para ambiente de nuvem computacional, o elastic-OrthoSearch também demonstrou resultados positivos relativos à aceleração dos experimentos frente às execuções serializadas do OrthoSearch. Em todos os cenários propostos, para todas as bases de ortólogos e organismos confrontados, foram obtidos valores entre 1,99 a 4,95 vezes mais rápidos do que nas execuções serializadas do OrthoSearch.

As bases de dados de ortólogos criadas através da metodologia proposta e integrada ao OrthoSearch permitiram elevar a oferta de grupos ortólogos de forma considerável. Especificamente a base mais extensa, “KO + EggNOG KOG + ProtozoaDB”, oferta 86,46% mais grupos ortólogos e 22,45% proteínas do que a

base KO, utilizada como ponto de partida para os experimentos associados à metodologia proposta nesta tese.

Foi possível analisar a pertinência de proteínas de protozoários, organismos de interesse especial do LBCS/IOC/Fiocruz, nos grupos ortólogos das bases criadas através desta metodologia.

Os resultados obtidos são encorajadores, com acréscimo de até 379% no quantitativo de grupos ortólogos com ao menos uma proteína de protozoário e com oferta de até 300% mais proteínas de protozoários através das bases criadas com a metodologia proposta.

No mesmo estudo realizado para a criação de tais bases também foi possível identificar 13 alvos potenciais em protozoários (*Leishmania* spp.).

Sendo assim, considera-se que todos os objetivos propostos nesta tese foram alcançados e que existe oportunidade para utilização direta das soluções aqui apresentadas, assim como para o desenvolvimento de melhorias para as mesmas.

# 7 REFERÊNCIAS BIBLIOGRÁFICAS


Aalst W van der, Hee KM van. *Workflow management: models, methods, and systems*. Cambridge, Mass: MIT Press; 2002.

Abouelhoda M, Issa SA, Ghanem M. *Tavaxy: Integrating Taverna and Galaxy workflows with cloud computing support*. **BMC Bioinformatics**. 4/5/2012;13:77.

Abrahamsen MS, Templeton TJ, Enomoto S, Abrahante JE, Zhu G, Lancto CA, et al. *Complete Genome Sequence of the Apicomplexan, Cryptosporidium parvum*. **Science**. 16/4/2004;304(5669):441–5.

Adam RD. *The Giardia lamblia genome*. **Int. J. Parasitol.** /Abril/2000;30(4):475–84.

Altintas I, Berkley C, Jaeger E, Jones M, Ludascher B, Mock S. *Kepler: an extensible system for design and execution of scientific workflows*. IEEE; 2004. p. 423–4. Disponível em: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1311241

Altschul SF, Gish W, Miller W, Myers EW, Lipman DJ. *Basic local alignment search tool*. **J. Mol. Biol.** 5/10/1990;215(3):403–10.

Amazon. *Amazon Elastic Compute Cloud (Amazon EC2)* [Internet]. 2012. Disponível em: http://aws.amazon.com/pt/ec2/

Amdahl GM. *Validity of the single processor approach to achieving large scale computing capabilities*. ACM Press; 1967. p. 483. Disponível em: http://portal.acm.org/citation.cfm?doid=1465482.1465560

Apache. *Apache Hadoop* [Internet]. 2013. Disponível em: http://hadoop.apache.org/

Armbrust M, Fox A, Griffith R, Joseph AD, Katz R, Konwinski A, et al. *A view of cloud computing*. **Commun ACM**. /4/2010;53(4):50–8.

Ashburner M, Ball CA, Blake JA, Botstein D, Butler H, Cherry JM, et al. *Gene Ontology: tool for the unification of biology*. **Nat. Genet.** /5/2000;25(1):25–9.

Aurrecoechea C, Brestelli J, Brunk BP, Carlton JM, Dommer J, Fischer S, et al. *GiardiaDB and TrichDB: integrated genomic resources for the eukaryotic protist pathogens Giardia lamblia and Trichomonas vaginalis*. **Nucleic Acids Res.** 1/1/2009;37(Database):D526–30.

Barrett MP, Burchmore RJ, Stich A, Lazzari JO, Frasch AC, Cazzulo JJ, et al. *The trypanosomiases*. **The Lancet**. /11/2003;362(9394):1469–80.

Bavoil L, Callahan SP, Crossno PJ, Freire J, Scheidegger CE, Silva CT, et al. *VisTrails: Enabling Interactive Multiple-View Visualizations*. IEEE; 2005. p. 135–42. Disponível em: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1532788

BiowebDB. *BiowebDB Consortium - ProtozoaDB - Development* [Internet]. 2012. Disponível em: http://protozoadb.biowebdb.org/

Brayton KA, Lau AOT, Herndon DR, Hannick L, Kappmeyer LS, Berens SJ, et al. *Genome sequence of Babesia bovis and comparative analysis of apicomplexan hemoprotozoa*. **PLoS Pathog.** 19/10/2007;3(10):1401–13.

Brotherton M-C, Bourassa S, Leprohon P, Légaré D, Poirier GG, Droit A, et al. *Proteomic and Genomic Analyses of Antimony Resistant Leishmania infantum Mutant*. Zilberstein D, editor. **PLoS ONE**. 27/11/2013;8(11):e81899.

Caballer M, Blanquer I, Moltó G, de Alfonso C. *Dynamic Management of Virtual Infrastructures*. **J. Grid Comput.** /3/2015;13(1):53–70.

Camacho C, Coulouris G, Avagyan V, Ma N, Papadopoulos J, Bealer K, et al. *BLAST+: architecture and applications*. **BMC Bioinformatics**. 2009;10:421.

Capella-Gutiérrez S. *get_representative_sequence* [Internet]. 2014. Disponível em: https://github.com/scapella/trimal/blob/dev/scripts/get_sequence_representative_from_alignment.py

Capella-Gutierrez S, Silla-Martinez JM, Gabaldon T. *trimAl: a tool for automated alignment trimming in large-scale phylogenetic analyses*. **Bioinformatics**. 1/8/2009;25(15):1972–3.

CAPES. *CAPES | PDSE | Programa de Doutorado Sanduíche no Exterior* [Internet]. 2015. Disponível em: http://www.capes.gov.br/bolsas/bolsas-no-exterior/programa-de-doutorado-sanduiche-no-exterior-pdse

Carlton JM, Angiuoli SV, Suh BB, Kooij TW, Pertea M, Silva JC, et al. *Genome sequence and comparative analysis of the model rodent malaria parasite Plasmodium yoelii yoelii*. **Nature**. 3/10/2002;419(6906):512–9.

Carlton JM, Hirt RP, Silva JC, Delcher AL, Schatz M, Zhao Q, et al. *Draft Genome Sequence of the Sexually Transmitted Pathogen Trichomonas vaginalis*. **Science**. 12/1/2007;315(5809):207–12.

Carrión A, Kotowski, Nelson, Caballer M, Blanquer I, Jardim R, Dávila AMR. *Design and implementation of a Generic and Multi-Platform Workflow System*. **8th Iber. Grid Infrastruct. Conf. Proc.** [Internet]. 2014. p. 77. Disponível em: http://riunet.upv.es/bitstream/handle/10251/43793/IBERGRID.%208th%20Iberian%20Grid%20Infrastructure%20Conference%20Proceedings_6177.pdf?sequence=1&isAllowed=y%22

Carrión JV, Moltó G, Alfonso CD, Caballer M, Hernández V. *A Generic Catalog and Repository Service for Virtual Machine Images*. 2010. Disponível em: http://cloudcomp.eu/2010/

Cavalier-Smith T. *Kingdom protozoa and its 18 phyla*. **Microbiol. Rev.** /12/1993;57(4):953–94.

Cavalier-Smith T. *Kingdoms Protozoa and Chromista and the eozoan root of the eukaryotic tree*. **Biol. Lett.** 23/12/2009a;:rsbl20090948.

Cavalier-Smith T. *Predation and eukaryote cell origins: A coevolutionary perspective*. **Int. J. Biochem. Cell Biol.** /Fevereiro/2009b;41(2):307–22.

CDC. *CDC | Leishmaniasis* [Internet]. CDC Leishmaniasis. 2015. Disponível em: http://www.cdc.gov/parasites/leishmaniasis/disease.html

CentOS. *CentOS* [Internet]. 2015. Disponível em: https://www.centos.org

Chen F, Mackey AJ, Stoeckert CJ, Roos DS. *OrthoMCL-DB: querying a comprehensive multi-species collection of ortholog groups*. **Nucleic Acids Res.** 1/1/2006;34(suppl 1):D363–8.

Chen F, Mackey AJ, Vermunt JK, Roos DS. *Assessing Performance of Orthology Detection Strategies Applied to Eukaryotic Genomes*. **PLoS ONE** [Internet]. 18/4/2007;2(4). Disponível em: http://www.ncbi.nlm.nih.gov/pmc/articles/PMC1849888/

Ciccarelli FD. *Toward Automatic Reconstruction of a Highly Resolved Tree of Life*. **Science**. 3/3/2006;311(5765):1283–7.

Colotti G, Baiocco P, Fiorillo A, Boffi A, Poser E, Chiaro FD, et al. *Structural insights into the enzymes of the trypanothione pathway: targets for antileishmaniasis drugs*. **Future Med. Chem.** /10/2013;5(15):1861–75.

Da Cruz SMS, Batista V, Dávila AMR, Silva E, Tosta F, Vilela C, et al. *OrthoSearch: a scientific workflow approach to detect distant homologies on protozoans*. New York, NY, USA: ACM; 2008. p. 1282–6. Disponível em: http://doi.acm.org/10.1145/1363686.1363983

Da Cruz SMS, Batista V, Silva E, Tosta F, Vilela C, Cuadrat R, et al. *Detecting distant homologies on protozoans metabolic pathways using scientific workflows*. **Int. J. Data Min. Bioinforma.** 2010;4(3):256–80.

Cuadrat RRC, Cruz SM da S, Tschoeke DA, Silva E, Tosta F, Jucá H, et al. *An Orthology-Based Analysis of Pathogenic Protozoa Impacting Global Health: An Improved Comparative Genomics Approach with Prokaryotes and Model Eukaryote Orthologs*. **OMICS J. Integr. Biol.** 24/6/2014;140624130015005.

Dalquen DA, Altenhoff AM, Gonnet GH, Dessimoz C. *The Impact of Gene Duplication, Insertion, Deletion, Lateral Gene Transfer and Sequencing Error on Orthology Inference: A Simulation Study*. **PLoS ONE**. /Fevereiro/2013;8(2):e56925.

Dávila AMR, Majiwa PAO, Grisard EC, Aksoy S, Melville SE. *Comparative genomics to uncover the secrets of tsetse and livestock-infective trypanosomes*. **Trends Parasitol.** /10/2003;19(10):436–9.

Dávila AMR, Mendes PN, Wagner G, Tschoeke DA, Cuadrat RRC, Liberman F, et al. *ProtozoaDB: dynamic visualization and exploration of protozoan genomes*. **Nucleic Acids Res.** /1/2008;36(Database issue):D547–52.

Dean J, Ghemawat S. *MapReduce: simplified data processing on large clusters*. **Commun ACM**. /1/2008;51(1):107–13.

Delsuc F, Brinkmann H, Philippe H. *Phylogenomics and the reconstruction of the tree of life*. **Nat. Rev. Genet.** /5/2005;6(5):361–75.

DeLuca TF, Cui J, Jung J-Y, Gabriel KCS, Wall DP. *Roundup 2.0: Enabling comparative genomics for over 1800 genomes*. **Bioinformatics** [Internet]. 13/1/2012; Disponível em: http://bioinformatics.oxfordjournals.org/content/early/2012/01/13/bioinformatics.bts006

Dessimoz C. *Editorial: Orthology and applications*. **Brief. Bioinform.** 1/9/2011;12(5):375–6.

Dessimoz C, Gabaldón T, Roos DS, Sonnhammer ELL, Herrero J, Quest for Orthologs Consortium. *Toward community standards in the quest for orthologs*. **Bioinforma. Oxf. Engl.** 15/3/2012;28(6):900–4.

DNDi. *DNDi | Drugs for Neglected Diseases initiative* [Internet]. 2015. Disponível em: http://www.dndi.org/about-us/overview-dndi/founding-partners.html

ECMA. *ECMA | The JSON Data Interchange Format* [Internet]. 2013. Disponível em: http://www.ecma-international.org/publications/files/ecma-st/ECMA-262.pdf

Elmore SA, Jones JL, Conrad PA, Patton S, Lindsay DS, Dubey JP. *Toxoplasma gondii: epidemiology, feline clinical aspects, and prevention*. **Trends Parasitol.** /4/2010;26(4):190–6.

El-On J. *Current status and perspectives of the immunotherapy of leishmaniasis*. **Isr. Med. Assoc. J. IMAJ**. /10/2009;11(10):623–8.

Enright AJ. *An efficient algorithm for large-scale detection of protein families*. **Nucleic Acids Res.** 1/4/2002;30(7):1575–84.

Finn RD, Clements J, Eddy SR. *HMMER web server: interactive sequence similarity searching*. **Nucleic Acids Res.** 1/7/2011;39(suppl):W29–37.

Flicek P, Amode MR, Barrell D, Beal K, Brent S, Carvalho-Silva D, et al. *Ensembl 2012*. **Nucleic Acids Res.** 1/1/2012;40(D1):D84–90.

Foster I, Zhao Y, Raicu I, Lu S. *Cloud Computing and Grid Computing 360-Degree Compared*. **Grid Comput. Environ. Workshop 2008 GCE 08**. 2008. p. 1–10.

Freire J, Koop D, Santos E, Silva C. *Provenance for Computational Tasks: A Survey*. **Comput. Sci. Eng.** /5/2008;10(3):11–21.

Gabriel E, Fagg GE, Bosilca G, Angskun T, Dongarra JJ, Squyres JM, et al. *Open MPI: Goals, Concept, and Design of a Next Generation MPI Implementation*. In: Kranzlmüller D, Kacsuk P, Dongarra J, editors. **Recent Adv. Parallel Virtual Mach. Message Passing Interface** [Internet]. Springer Berlin Heidelberg; 2004. p. 97–104. Disponível em: http://link.springer.com/chapter/10.1007/978-3-540-30218-6_19

Garami A, Ilg T. *The role of phosphomannose isomerase in Leishmania mexicana glycoconjugate synthesis and virulence*. **J. Biol. Chem.** 2/3/2001;276(9):6566–75.

Goecks J, Nekrutenko A, Taylor J, Galaxy Team T. *Galaxy: a comprehensive approach for supporting accessible, reproducible, and transparent computational research in the life sciences*. **Genome Biol.** 2010;11(8):R86.

Grimaldi G, Tesh RB. *Leishmaniases of the New World: current concepts and implications for future research.* **Clin. Microbiol. Rev.** 1/7/1993;6(3):230–50.

Gurtowski J, Schatz MC, Langmead B. *Genotyping in the Cloud with Crossbow*. In: Baxevanis AD, Petsko GA, Stein LD, Stormo GD, editors. **Curr. Protoc. Bioinforma.** [Internet]. Hoboken, NJ, USA: John Wiley & Sons, Inc.; 2012. Disponível em: http://doi.wiley.com/10.1002/0471250953.bi1503s39

GUS. *GUS: The Genomics Unified Schema* [Internet]. 2012. Disponível em: http://www.gusdb.org/index.php

Haldar K, Mohandas N. *Malaria, erythrocytic infection, and anemia*. **ASH Educ. Program Book**. 1/1/2009;2009(1):87–93.

Hall N, Carlton J. *Comparative genomics of malaria parasites*. **Curr. Opin. Genet. Dev.** /12/2005;15(6):609–13.

Hardison RC. *Comparative Genomics*. **PLoS Biol.** /11/2003;1(2):156–60.

Hiramatsu T, Kodama K, Kuroda T, Mizushima T, Tsuchiya T. *A putative multisubunit Na+/H+ antiporter from Staphylococcus aureus*. **J. Bacteriol.** /12/1998;180(24):6642–8.

Höfer CN, Karagiannis G. *Cloud computing services: taxonomy and comparison*. **J. Internet Serv. Appl.** /9/2011;2(2):81–94.

Huang C, Turco SJ. *Defective galactofuranose addition in lipophosphoglycan biosynthesis in a mutant of Leishmania donovani*. **J. Biol. Chem.** 15/11/1993;268(32):24060–6.

Hunfeld K-P, Hildebrandt A, Gray JS. *Babesiosis: Recent insights into an ancient disease*. **Int. J. Parasitol.** /9/2008;38(11):1219–37.

Imam TS. *The complexities in the classification of protozoa: a challenge to parasitologists*. **Bayero J. Pure Appl. Sci.** 2009;2(2):159–64.

Ito M, Guffanti AA, Krulwich TA. *Mrp-dependent Na+/H+ antiporters of Bacillus exhibit characteristics that are unanticipated for completely secondary active transporters*. **FEBS Lett.** /5/2001;496(2-3):117–20.

Jardim R. *Estudo de reposicionamento de fármacos para doenças negligenciadas causadas por protozoários através da integração de bases de dados biológicas usando Web Semântica* [Internet]. Instituto Oswaldo Cruz; 2013. Disponível em: http://arca.icict.fiocruz.br/handle/icict/7027

Jensen LJ, Julien P, Kuhn M, von Mering C, Muller J, Doerks T, et al. *eggNOG: automated construction and annotation of orthologous groups of genes*. **Nucleic Acids Res.** /1/2008;36(Database issue):D250–4.

Jones TW, Dávila AMR. *Trypanosoma vivax – out of Africa*. **Trends Parasitol.** 1/2/2001;17(2):99–101.

Joosten S, Brinkkemper S. *Fundamental Concepts for Workflow Automation in Practice*. Stockholm; 1995. p. 311–2.

Kanehisa M, Goto S, Sato Y, Furumichi M, Tanabe M. *KEGG for integration and interpretation of large-scale molecular data sets*. **Nucleic Acids Res.** 1/1/2012;40(D1):D109–14.

Katoh K, Standley DM. *MAFFT Multiple Sequence Alignment Software Version 7: Improvements in Performance and Usability*. **Mol. Biol. Evol.** 1/4/2013;30(4):772–80.

Kegg Orthology K. *KEGG Orthology* [Internet]. Kegg Orthology. 2012. Disponível em: http://www.genome.jp/kegg/ko.html

Killick-Kendrick R, Peters W, Killick-Kendrick R, Peters W. *Rodent malaria.* 1978; Disponível em: http://www.cabdirect.org/abstracts/19792900680.html;jsessionid=6DA640E2EBEF812EEF71BDCBCEE1828C?freeview=true

Konstantinidis KT, Tiedje JM. *Genomic insights that advance the species definition for prokaryotes*. **Proc. Natl. Acad. Sci.** 15/2/2005;102(7):2567–72.

Koonin E, Galperin M. *Sequence - Evolution - Function - NCBI Bookshelf* [Internet]. 2003. Disponível em: http://www.ncbi.nlm.nih.gov/books/NBK20260/

Koonin EV. *Orthologs, Paralogs, and Evolutionary Genomics*. **Annu. Rev. Genet.** 2005;39(1):309–38.

Kotowski N, Jardim R, Dávila AMR. *Improved orthologous databases to ease protozoan targets inference*. **Parasit. Vectors** [Internet]. /12/2015;8(1). Disponível em: http://www.parasitesandvectors.com/content/8/1/494

Kotpal RL. *Modern text book of zoology: invertebrates (animal diversity - 1)*. Meerut: Rastogi Publications; 2012.

Krauth-Siegel RL, Meiering SK, Schmidt H. *The Parasite-Specific Trypanothione Metabolism of Trypanosoma and Leishmania*. **Biol. Chem.** [Internet]. 10/1/2003;384(4). Disponível em: http://www.degruyter.com/view/j/bchm.2003.384.issue-4/bc.2003.062/bc.2003.062.xml

Kristensen DM, Wolf YI, Mushegian AR, Koonin EV. *Computational methods for Gene Orthology inference*. **Brief. Bioinform.** /9/2011;12(5):379–91.

Langmead B, Schatz MC, Lin J, Pop M, Salzberg SL. *Searching for SNPs with cloud computing*. **Genome Biol.** 2009a;10(11):R134.

Langmead B, Trapnell C, Pop M, Salzberg SL. *Ultrafast and memory-efficient alignment of short DNA sequences to the human genome*. **Genome Biol.** 2009b;10(3):R25.

Lassmann T, Sonnhammer ELL. *Kalign--an accurate and fast multiple sequence alignment algorithm*. **BMC Bioinformatics**. 2005;6:298.

Lewis MD, Ma J, Yeo M, Carrasco HJ, Llewellyn MS, Miles MA. *Genotyping of Trypanosoma cruzi: Systematic Selection of Assays Allowing Rapid and Accurate Discrimination of All Known Lineages*. **Am. J. Trop. Med. Hyg.** 1/12/2009;81(6):1041–9.

Li L, Stoeckert CJ, Roos DS. *OrthoMCL: Identification of Ortholog Groups for Eukaryotic Genomes*. **Genome Res.** 1/9/2003;13(9):2178–89.

Li R, Li Y, Fang X, Yang H, Wang J, Kristiansen K, et al. *SNP detection for massively parallel whole-genome resequencing*. **Genome Res.** 1/6/2009;19(6):1124–32.

Lorenzi HA, Puiu D, Miller JR, Brinkac LM, Amedeo P, Hall N, et al. *New Assembly, Reannotation and Analysis of the Entamoeba histolytica Genome Reveal New Genomic Features and Protein Content Information*. **PLoS Negl Trop Dis**. 15/6/2010;4(6):e716.

Ludäscher B, Altintas I, Berkley C, Higgins D, Jaeger E, Jones M, et al. *Scientific workflow management and the Kepler system*. **Concurr. Comput. Pract. Exp.** 25/8/2006;18(10):1039–65.

Matsunaga A, Tsugawa M, Fortes J. *CloudBLAST: Combining MapReduce and Virtualization on Distributed Resources for Bioinformatics Applications*. **IEEE Fourth Int. Conf. EScience 2008 EScience 08**. 2008. p. 222–9.

Mattoso M, Werner C, Travassos GH, Braganholo V, Ogasawara E, Oliveira DD, et al. *Towards supporting the life cycle of large scale scientific experiments*. **Int. J. Bus. Process Integr. Manag.** 2010;5(1):79.

McCormack JE, Faircloth BC, Crawford NG, Gowaty PA, Brumfield RT, Glenn TC. *Ultraconserved elements are novel phylogenomic markers that resolve placental mammal phylogeny when combined with species-tree analysis*. **Genome Res.** 1/4/2012;22(4):746–54.

Mell P, Grance T. *The NIST Definition of Cloud Computing* [Internet]. 2011 Sep. Report No.: 800-145. Disponível em: http://faculty.winthrop.edu/domanm/csci411/Handouts/NIST.pdf

Miller JA, Sheth AP, Kochut KJ, Xuzhong Wang, Murugan A. *Simulation modeling within workflow technology*. IEEE; 1995. p. 612–9. Disponível em: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=478807

MongoDB. *MongoDB* [Internet]. 2015. Disponível em: https://www.mongodb.org/

Morrison HG, McArthur AG, Gillin FD, Aley SB, Adam RD, Olsen GJ, et al. *Genomic Minimalism in the Early Diverging Intestinal Parasite Giardia lamblia*. **Science**. 28/9/2007;317(5846):1921–6.

MPI. *MPI* [Internet]. 2013. Disponível em: http://www.dmoz.org/Computers/Parallel_Computing/Programming/Libraries/MPI/

MSF. *Fatal Imbalance - The Crisis in Research and Development for Drugs for Neglected Diseases* [Internet]. Daniel Berman, Suerie Moon; 2001. Disponível em: http://www.msfaccess.org/our-work/neglected-diseases/article/958

Murray HW, Berman JD, Davies CR, Saravia NG. *Advances in leishmaniasis*. **The Lancet**. 29/10/2005;366(9496):1561–77.

NCBI. *RefSeq: NCBI Reference Sequence Database* [Internet]. 2015. Disponível em: http://www.ncbi.nlm.nih.gov/refseq/

NIST. *NIST | Cloud Computing* [Internet]. 2015. Disponível em: http://www.nist.gov/itl/cloud/

Nurmi D, Wolski R, Grzegorczyk C, Obertelli G, Soman S, Youseff L, et al. *The Eucalyptus Open-Source Cloud-Computing System*. IEEE; 2009. p. 124–31. Disponível em: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5071863

Oinn T, Addis M, Ferris J, Marvin D, Senger M, Greenwood M, et al. *Taverna: a tool for the composition and enactment of bioinformatics workflows*. **Bioinformatics**. 22/11/2004;20(17):3045–54.

Oliveira D. *Uma abordagem de apoio à execução paralela de "workflows" científicos em nuvens de computadores* [Internet]. UFRJ; 2012. Disponível em: http://146.164.2.115/F/AJF8YHKF76VF8K27FPVAGM1398RRE1MTFMGBHJDVXKKUQRVYQ9-38482?func=item-global&doc_library=UFR01&doc_number=000789364&year=&volume=&sub_library=

De Oliveira D, Ogasawara E, Baião F, Mattoso M. *SciCumulus: A Lightweight Cloud Middleware to Explore Many Task Computing Paradigm in Scientific Workflows*. IEEE; 2010. p. 378–85. Disponível em: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5557969

Ollomo B, Durand P, Prugnolle F, Douzery E, Arnathau C, Nkoghe D, et al. *A New Malaria Agent in African Hominids*. **PLoS Pathog**. 29/5/2009;5(5):e1000446.

OpenNebula. *OpenNebula* [Internet]. 2015. Disponível em: http://opennebula.org

OpenStack. *OpenStack* [Internet]. 2010. Disponível em: http://www.openstack.org

OrthoMCL. *OrthoMCL* [Internet]. 2015. Disponível em: http://www.orthomcl.org/orthomcl/showQuestion.do;jsessionid=60A6660A52E9B119CCE3F538520C43DB?questionFullName=GroupQuestions.ByPhyleticPattern

Pain A, Böhme U, Berry AE, Mungall K, Finn RD, Jackson AP, et al. *The genome of the simian and human malaria parasite Plasmodium knowlesi*. **Nature**. 9/10/2008;455(7214):799–803.

Pain A, Renauld H, Berriman M, Murphy L, Yeats CA, Weir W, et al. *Genome of the host-cell transforming parasite Theileria annulata compared with T. parva*. **Science**. 1/7/2005;309(5731):131–3.

Pearson WR. *Rapid and sensitive sequence comparison with FASTP and FASTA*. **Methods Enzymol.** [Internet]. Elsevier; 1990. p. 63–98. Disponível em: http://ukpmc.ac.uk/abstract/MED/2156132/reload=0;jsessionid=Za5NHoS47bc0YDeUxeYf.4

Powell S, Forslund K, Szklarczyk D, Trachana K, Roth A, Huerta-Cepas J, et al. *eggNOG v4.0: nested orthology inference across 3686 organisms*. **Nucleic Acids Res.** 1/12/2013;gkt1253.

Powell S, Szklarczyk D, Trachana K, Roth A, Kuhn M, Muller J, et al. *eggNOG v3.0: orthologous groups covering 1133 organisms at 41 different taxonomic ranges*. **Nucleic Acids Res.** 16/11/2011;40(D1):D284–9.

Rasmussen MD, Kellis M. *A Bayesian Approach for Fast and Accurate Gene Tree Reconstruction*. **Mol. Biol. Evol.** 1/1/2011;28(1):273–90.

Remm M, Storm CEV, Sonnhammer ELL. *Automatic clustering of orthologs and in-paralogs from pairwise species comparisons*. **J. Mol. Biol.** 14/12/2001;314(5):1041–52.

Rodgers DP. *Improvements in Multiprocessor System Design*. **Proc. 12th Annu. Int. Symp. Comput. Archit.** [Internet]. Los Alamitos, CA, USA: IEEE Computer Society Press; 1985. p. 225–31. Disponível em: http://dl.acm.org/citation.cfm?id=327010.327215

R Project. *R Project* [Internet]. 2015. Disponível em: http://www.r-project.org

Ruberg N, Kotowski N, Mattos A, Matos L, Machado M, Oliveira D, et al. *Experiencing Data Grids*. In: Daydé M, Palma JMLM, Coutinho ÁLGA, Pacitti E, Lopes JC, editors. **High Perform. Comput. Comput. Sci. - VECPAR 2006** [Internet]. Berlin, Heidelberg: Springer Berlin Heidelberg; 2007. p. 707–18. Disponível em: http://link.springer.com/10.1007/978-3-540-71351-7_56

Sachs J, Malaney P. *The economic and social burden of malaria*. **Nature**. 7/2/2002;415(6872):680–5.

Shanker A. *Genome research in the cloud*. **Omics J. Integr. Biol.** /7/2012;16(7-8):422–8.

Shinde S, Mol M, Jamdar V, Singh S. *Molecular modeling and molecular dynamics simulations of GPI 14 in Leishmania major: insight into the catalytic site for active site directed drug design*. **J. Theor. Biol.** 21/6/2014;351:37–46.

Singh N, Chikara S, Sundar S. *SOLiD™ Sequencing of Genomes of Clinical Isolates of Leishmania donovani from India Confirm Leptomonas Co-Infection and Raise Some Key Questions*. Shomron N, editor. **PLoS ONE**. 13/2/2013;8(2):e55738.

Sonnhammer E. *Stockholm format* [Internet]. 2015. Disponível em: http://sonnhammer.sbc.su.se/Stockholm.html

Sorek R, Zhu Y, Creevey CJ, Francino MP, Bork P, Rubin EM. *Genome-Wide Experimental Determination of Barriers to Horizontal Gene Transfer*. **Science**. 30/11/2007;318(5855):1449–52.

De Souza W, Kritski AL, Morel CM, de Lemos ERS, Garcia E, Camargo EP, et al. *Doenças Negligenciadas* [Internet]. Academia Brasileira de Ciências; 2010. Disponível em: http://www.abc.org.br/IMG/pdf/doc-199.pdf

Spath GF, Epstein L, Leader B, Singer SM, Avila HA, Turco SJ, et al. *Lipophosphoglycan is a virulence factor distinct from related glycoconjugates in the protozoan parasite Leishmania major*. **Proc. Natl. Acad. Sci.** 1/8/2000;97(16):9258–63.

Stein LD. *The case for cloud computing in genome informatics*. **Genome Biol.** 5/5/2010;11(5):207.

Tatusov RL, Fedorova ND, Jackson JD, Jacobs AR, Kiryutin B, Koonin EV, et al. *The COG database: an updated version includes eukaryotes*. **BMC Bioinformatics**. 11/9/2003;4:41.

Taylor IJ, Deelman E, Gannon DB, Shields M. *Workflows for e-Science* [Internet]. 2007. Disponível em: http://www.springer.com/computer/communication+networks/book/978-1-84628-519-6

Taylor RC. *An overview of the Hadoop/MapReduce/HBase framework and its current applications in bioinformatics*. **BMC Bioinformatics**. 21/12/2010;11(Suppl 12):S1.

Tersteegen A, Hedderich R. *Methanobacterium thermoautotrophicum encodes two multisubunit membrane-bound [NiFe] hydrogenases. Transcription of the operons and sequence analysis of the deduced proteins*. **Eur. J. Biochem. FEBS**. /9/1999;264(3):930–43.

The MPI Forum C. *MPI: a message passing interface*. **Proc. 1993 ACMIEEE Conf. Supercomput.** [Internet]. New York, NY, USA: ACM; 1993. p. 878–83. Disponível em: http://doi.acm.org/10.1145/169627.169855

The UniProt Consortium. *Update on activities at the Universal Protein Resource (UniProt) in 2013*. **Nucleic Acids Res.** 1/1/2013;41(D1):D43–7.

Thompson RCA. *The impact of Giardia on science and society.* In: Ortega-Pierres G, Cacciò S, Fayer R, Mank TG, Smith HV, Thompson RCA, editors. **Giardia Cryptosporidium Mol. Dis.** [Internet]. Wallingford: CABI; 2009. p. 1–11. Disponível em: http://www.cabi.org/cabebooks/ebook/20093086272

Tschoeke D. *Genômica Comparativa de Protozoários*. Instituto Oswaldo Cruz; 2013.

Tschoeke DA, Nunes GL, Jardim R, Lima J, Dumaresq AS, Gomes MR, et al. *The Comparative Genomics and Phylogenomics of Leishmania amazonensis Parasite*. **Evol. Bioinforma. Online**. 23/9/2014;10:131–53.

Uniting to Combat NTDs. *The London declaration on Neglected Tropical Diseases* [Internet]. 2014. Disponível em: http://unitingtocombatntds.org/resource/london-declaration

Vallejo GA, Guhl F, Carranza JC, Lozano LE, Sánchez JL, Jaramillo JC, et al. *kDNA markers define two major Trypanosoma rangeli lineages in Latin-America*. **Acta Trop.** /1/2002;81(1):77–82.

Wall DP, Deluca T. *Ortholog detection using the reciprocal smallest distance algorithm*. **Methods Mol. Biol. Clifton NJ**. 2007;396:95–110.

Wall DP, Fraser HB, Hirsh AE. *Detecting putative orthologs*. **Bioinformatics**. 1/9/2003;19(13):1710–1.

Wall DP, Kudtarkar P, Fusaro VA, Pivovarov R, Patil P, Tonellato PJ. *Cloud computing for comparative genomics*. **BMC Bioinformatics**. 18/5/2010;11(1):259.

Weiss LM, Dubey JP. *Toxoplasmosis: A history of clinical observations*. **Int. J. Parasitol.** 1/7/2009;39(8):895–901.

Wellems TE, Hayton K, Fairhurst RM. *The impact of malaria parasitism: from corpuscles to communities*. **J. Clin. Invest.** 1/9/2009;119(9):2496–505.

WFMC. *Workflow Management Coalition | Terminology & Glossary TC-1011* [Internet]. 1999. Disponível em: http://www.wfmc.org/docs/TC-1011_term_glossary_v3.pdf

WFMC. *Workflow Management Coalition | WfMC* [Internet]. 2015. Disponível em: http://www.wfmc.org

WHO. *WHO | Research and Development* [Internet]. 2010a. Disponível em: http://www.who.int/trade/glossary/story079/en/

WHO. *WHO | Chagas disease (American trypanosomiasis)* [Internet]. WHO. 2010b. Disponível em: http://www.who.int/mediacentre/factsheets/fs340/en/index.html

WHO. *WHO | Neglected Diseases* [Internet]. WHO. 2012a. Disponível em: http://www.who.int/neglected_diseases/diseases/en/

WHO. *WHO | Human African trypanosomiasis (sleeping sickness)* [Internet]. WHO. 2012b. Disponível em: http://www.who.int/mediacentre/factsheets/fs259/en/

WHO. *WHO | TDR | Diseases and topics* [Internet]. 2014. Disponível em: http://www.who.int/tdr/diseases-topics/en/

WHO. *Investing to Overcome the Global Impact of Neglected Tropical Diseases.* World Health Organization; 2015a.

WHO. *WHO | Leishmaniasis* [Internet]. WHO. 2015b. Disponível em: http://www.who.int/leishmaniasis/en/

WHO. *WHO | Leishmaniasis Fact Sheet* [Internet]. WHO. 2015c. Disponível em: http://www.who.int/mediacentre/factsheets/fs375/en/

Widmer G, London E, Zhang L, Ge G, Tzipori S, Carlton JM, et al. *Preliminary analysis of the Cryptosporidium muris genome.* In: Ortega-Pierres G, Cacciò S, Fayer R, Mank TG, Smith HV, Thompson RCA, editors. **Giardia Cryptosporidium Mol. Dis.** [Internet]. Wallingford: CABI; 2009. p. 320–7. Disponível em: http://www.cabi.org/cabebooks/ebook/20093086297

XML. *XML | eXtensible Markup Language*. 2008.

Xu P, Widmer G, Wang Y, Ozaki LS, Alves JM, Serrano MG, et al. *The genome of Cryptosporidium hominis*. **Nature**. 28/10/2004;431(7012):1107–12.

Yang Z. *PAML 4: Phylogenetic Analysis by Maximum Likelihood*. **Mol. Biol. Evol.** 18/4/2007;24(8):1586–91.

Zhang K, Barron T, Turco SJ, Beverley SM. *The LPG1 gene family of Leishmania major*. **Mol. Biochem. Parasitol.** /7/2004;136(1):11–23.

# 8 ANEXOS

## 8.1 Anexo A - Artigo 1 – "Design and implementation of a Generic and Multi-Platform Workflow System"

Carrión A, **Kotowski N**, Caballer M, Blanquer I, Jardim R, Davila AMR. **Design and implementation of a Generic and Multi-Platform Workflow System.**

**Este artigo foi publicado e apresentado na conferência 8th Ibergrid Conference (Iberian Grid Infrastructure Conference), em Aveiro, Portugal, em setembro de 2014.**

***Design and implementation of a Generic and Multi-Platform Workflow System***

Data da Publicação: 19 de Setembro de 2014

Autores: Carrión, Abel; Caballer, Miguel; Blanquer, Ignacio; Kotowski, Nelson; Jardim, Rodrigo; Dávila, Alberto;

ISBN: 978-84-9048-246-9.

Disponível em: https://bioinformatics.ua.pt/ibergrid2014/proceedings/

Este foi o primeiro artigo submetido em parceria com o I3M/UPV e está relacionado para com os objetivos específicos 2.2.2 e 2.2.3. Neste momento, foi apresentada a versão conceitual da máquina de *workflows* que seria construída para possibilitar a execução do elastic-OrthoSearch em ambiente computacional distribuído. Para isso, foi necessário especificar um caso de uso com o OrthoSearch e entender como cada tarefa do mesmo seria integrada à referida máquina.

# Design and implementation of a Generic and Multi-Platform Workflow System

Abel Carrión[1], Nelson Kotowski[2,**], Miguel Caballer[1],
Ignacio Blanquer[1], Rodrigo Jardim[2], and Alberto MR Dávila[2]

[1]Instituto de Instrumentación para Imagen Molecular (I3M).
Centro mixto CSIC - Universitat Politècnica de València - CIEMAT
Camino de Vera s/n, 46022 Valencia, Spain
{abcarcol,micafer1,iblanquer}@upv.es
[2]Laboratório de Biologia Computacional e Sistemas, Instituto Oswaldo Cruz (IOC),
Fundação Oswaldo Cruz (FIOCRUZ), Avenida Brasil 4365,
21040-360, Rio de Janeiro, Rio de Janeiro, Brazil
{nelsonpeixoto,rodrigo_jardim,davila}@fiocruz.br

**Abstract.** Nowadays e-Science experiments require computational and storage resources that most research centres cannot afford. Fortunately, researchers have at their disposal many distributed computing platforms where to run their experiments: clusters, supercomputers, grids and clouds. None of these platforms has showed to be the ideal choice and each of them has its own advantages and disadvantages, depending on various factors. However, the use of these powerful systems poses a challenge for scientists who don't have a computer science background. For that reason, this paper describes the design and implementation of an intuitive workflow system capable of executing any kind of application on a mix of computing platforms. Moreover, a bioinformatics use case called Ortosearch that will exploit the workflow system is detailed.

## 1 Introduction

In the business context, the term workflow can be defined as the orchestration of a set of activities in order to accomplish a larger and sophisticated goal. A specialization of this term can be found on e-Science, the Scientific Workflows (SWFs). SWFs are a formalization of the particular process that a scientist must carry on to obtain publishable results from raw data. SWFs can also be seen as a collection of tasks that are processed on computing resources in a well-defined order to accomplish a goal. Due to the high computational cost of e-Science experiments, the computing resources needed are no longer localized but, rather, distributed and hence, developers of scientific applications (workflows) have many options when choosing the platform where to run their applications. In the last decade, these options included: clusters, supercomputers and Grid. Grid computing focuses on secure and collaborative resource sharing across multiple, geographically distributed institutions. One of the main drawbacks that Grid users face is that while

** CAPES/PDSE scholarship BEX 2137/14-3

a Grid offers access to many heterogeneous resources, a user normally needs a very specific environment that is customized to support a particular legacy application. Obviously, resource providers cannot support the diversity of all the required environments. In the last years, Cloud Computing has emerged as another viable [1] platform for running scientific applications. In particular, the use of virtualization provides many useful benefits for scientific workflow applications, including: customization of the software stack by the user, performance isolation, check-pointing and migration, better reproducibility of scientific analyses, and enhanced support for legacy applications [2] [3]. However, in other to compete with existing HPC systems in terms of performance, Cloud providers must begin offering high-speed networks and parallel file systems [4].
Because each platform has its own advantages and disadvantages in terms of usability, performance and cost, this work exposes the design and implementation of a generic (from the point of view of the applications that can be used) and multiplatform workflow system. One of the main concerns is to ease the combined use of the before mentioned platforms for executing SWFs, even for scientists with little knowledge about computer science.
The remainder of this paper is structured as follows. Firstly, Section 2 offers a brief description of similar tools and remarks the differences with respect the work presented in this paper. Next, Section 3 details the design of the architecture of the tool and Section 4 its correspondent implementation. As use case, Section 5 shows a bioinformatics workflow called Ortosearch that will exploit this work. To wrap up, the most relevant conclusions are extracted and the imminent work to do is proposed.

## 2 State-of-the-art

The use of workflow systems for executing scientific experiments is a topic widely covered in the literature. In this section, the criteria adopted for classifying these solutions has been: firstly, its generality (i.e. if it addresses any kind of application or only applications of a concrete field) and secondly, the distributed computing infrastructures supported.
Thus, the first category to be considered includes the generic solutions. With the rise of the Grid Computing paradigm, a large number of tools were proposed and used by the Grid Community. ASKALON [5] simplified the execution of workflow applications on the Grid with and XML-based programming interface that hides Grid middleware details to the user and allows the high-level composition of workflow applications. GridAnt [6] is an extensible and platform-independent workflow system that allows orchestrating a set of activities and expressing dependencies between them in XML. GridFlow [7] provides an user portal and an agent-based mechanism for dynamic scheduling of Grid jobs. Karajan is a workflow-based graphical problem solving environment which is effective at composing jobs in a complex Grid environment. The Kepler project [8] provides a workflow system derived from the Ptolomey II project. In this project a workflow system is modelled as a composition of actors (that are independent). The extensibility of this

system is given by the actor-oriented architecture, adding new actors. The system has a set of actors for performing typical Grid operations (authentication, file copy, job execution, job monitoring, service discovery, etc.). Nevertheless, the user needs to know a lot of details about the Grid resources. Pegasus [9] is a popular framework for mapping scientific workflows onto distributed resources. It uses an abstract workflow definition (users does not need to know details about the beneath systems) and workflow restructuring that clusters tasks in order to improve the overall performance. Webflow and Triana [10] offer a visual programming model for dynamic orchestration of high performance applications from a group of predefined commodity software modules. Both are powerful visual programming tools and enablers for Grid technology, abstracting the scientist from computer science details. When the Cloud Computing appeared, some tools from the Grid period were updated for supporting this new paradigm while others were developed anew. A good example of a Cloud-updated tool is Taverna [11]. Taverna is the workflow management system of the myGrid project. It is based in a definition language named Simple Conceptual Unified Flow Language (SCUFL). Taverna provides data models, enactor task executions, and graphical user interfaces. In Taverna, all computational workflow steps are web services. The introduction of the Taverna Server allows workflows to be executed on remote computational infrastructures (such as clusters, Grids and clouds). On the other hand, a project named e-Science Central (e-SC) [12] has developed a cloud-based Science Platform that allows scientists to store, analyse and share data in the cloud. e-SC can be deployed on both private and public Clouds. The CARMEN e-science project has also designed a generic e-science platform that runs in the cloud (in particular, Amazon AWS) and enables data sharing, integration and analysis. Although these tools are multi-platform, the workflow must be executed entirely in one of them, without the possibility of combining resources from different types of infrastructures. Moreover, when using Cloud Computing the virtual machines are statically deployed (previously to the execution), instead on-demand or dynamically as the NIST Cloud Computing definition says.

The second category entails non-generic systems, in concrete solutions related with the scientific field of the use case presented in this paper: bioinformatics. Galaxy [13] is an open web-based platform for genomic research, that makes computation accessible, ensures that all analysis are reproducible and allows the transparency via the sharing of experimental results between users. A Galaxy instance can utilize compute clusters for running jobs, and can be easily interfaced with PBS (Portable Batch System) and SGE (Sun Grid Engine). Chipster [14] offers a wide collection of data analysis methods within the reach of bioscientists via an intuitive graphical user interface. The analysis options are complemented with interactive visualizations and the possibility of saving workflows, which can be shared with other users.

The main difference between all the previous solutions and the one proposed in this work is that it comprises the following features in a single tool:

1. **Platform-agnostic client**: The client has been developed using a platform-independent programming language.

2. **Easy-to-use**: Although still not developed, the final version will offer a rich and usable Graphic User Interface (GUI).
3. **Generic**: The workflow system admits any kind of application that can be defined using its Workflow Specification Language.
4. **Multi-platform**: *Each part* of the workflow can be executed using different computing back-ends (clusters, supercomputers, Grids and Clouds).
5. **Cloud Computing features**: Meets the requirements expressed by the NIST Cloud Computing definition (dynamic provisioning of resources among others).

# 3 Architecture

The purpose of this section is twofold: firstly, to describe the design principles that guided the definition of the architecture and secondly, to show an overview of the architecture of the tool.

## 3.1 Design principles

The architecture has been designed taking into account three principles:

1. **Generality**: Possibility of executing a wide range of applications and supported by any kind of distributed computing infrastructure.
2. **Extensibility**: Enables a user to add a new application or a new computing back-end.
3. **Modularity**: Allows that new elements can be added without the need to change other parts of the system.

## 3.2 Overview

The architecture schema shown in Figure 1 is inspired by the one showed in [15], but updated accordingly to include the Cloud Computing paradigm. Basically, the architecture is divided into three sections: Workflow Design and Definition, Workflow Execution and Control and Interaction with the Resources. In turn, each of these three sections contains the components of the architecture. 'Workflow Design & Definition' is related with the Workflow Application Modelling & Definition Tools and the deployment time part of the Workflow Specification. 'Workflow Execution & Control' is the core of the architecture and includes the run time part of the Workflow Specification and the Workflow Enactment Service. Finally, the interaction with the computing and data resources is done through a set of connectors (plug-ins) for the different computing and storage back-ends.

# 4 Implementation

Once showed the architecture in the previous point, this section describes what technologies and/or software components have been used in the implementation of each element of the architecture. The system has been mostly implemented on Java because it is a platform-independent language and so, it can reach a wider number of users.

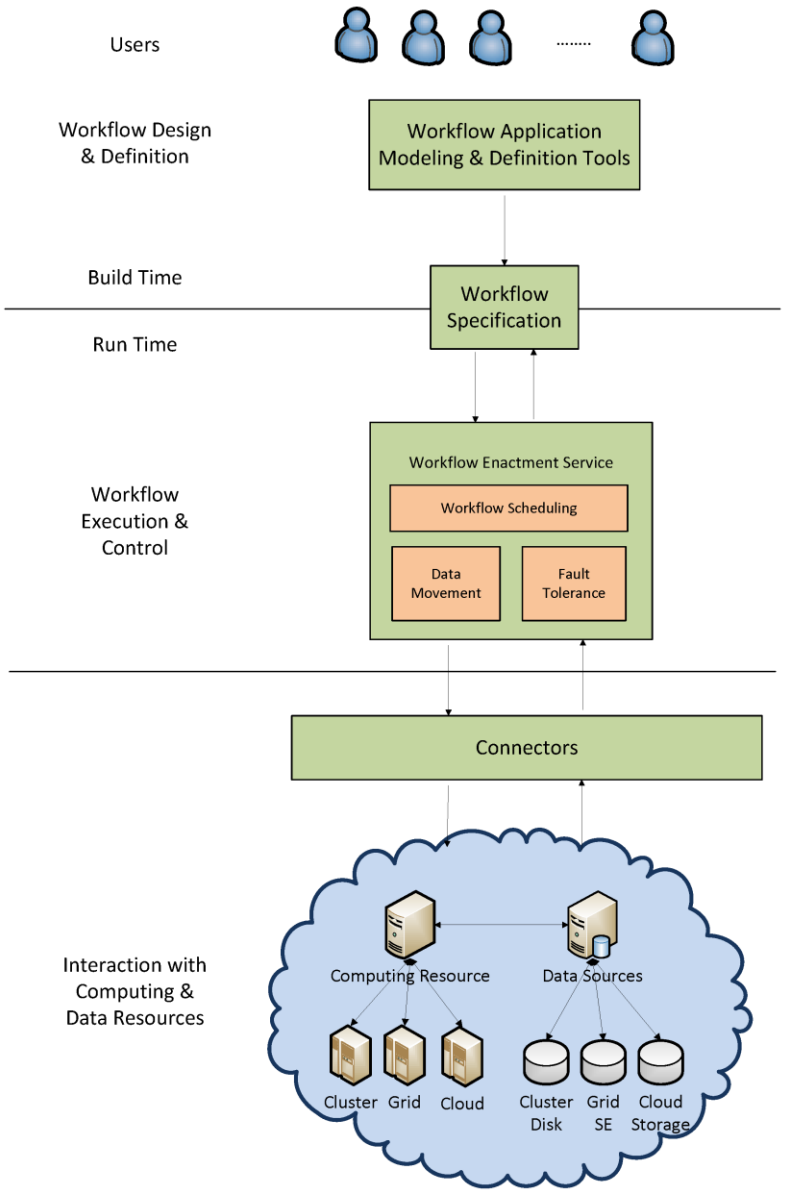


**Fig. 1.** Architecture overview

### 4.1 GUI

As many other workflow tools, in a future version the system will have a graph-base editing environment. Users compose applications by dragging programming components, called units or tools, from toolboxes, and dropping them onto a scratch pad, or workspace. Connectivity between the units is achieved by drawing cables subject to type-checking.
A workflow concrete specification is then generated by this visual tool and passed to the workflow planner. In order to improve the usability of the tool, this process is transparent to the user.

### 4.2 Workflow specification language

The workflow specification language is designed to bridge the gap between the GUI and the workflow execution engine. The format chosen for describing the workflows has been JSON (JavaScript Object Notation) because it has some benefits over XML, such as: shorter, easier to write and read and does not use end tags. The basic structure of a workflow specification document comprises three main sections (or objects in the JSON context): 'hosts', 'environments' and 'stages'. 'Hosts' is the part of the document that contains all the information about the infrastructure front-ends to be used. 'Environments' is a field for selecting the characteristics of the nodes and the packages that they must contain. Allows selecting nodes on heterogeneous environments and defining the Virtual Appliances (VAs) in a Cloud Computing scenario. Last but not least, Stages is an array of JSON objects that describes the different stages of the workflow. In turn, a stage object must: reference a host from the list of 'Hosts', include the features of the VM in the case of the Cloud (number of nodes, memory size, number of disks and capacity of each of them), list the command-lines that the stage executes, indicate fault-tolerance parameters and describe the stage-ins and stage-outs of the stage.
Currently, because the GUI component is under development, the entry point to the system is the JSON document. Once defined, the JSON document is passed to Jackson (a popular JSON parser) for parsing and validation and if it all is correct the workflow information is dumped to the program memory as Java objects.

### 4.3 Planner

The process of mapping from the concrete to the executable workflow can be automated by the planner. During that mapping the original workflow undergoes a series of refinements geared towards transforming the workflow to an executable description and towards optimizing the performance of the overall application. Figure 2 shows the three conversions that the planner performs. The first refinement (a) fuses two o more stages when the following two conditions are given: firstly, all of them are executed on the same infrastructure with the same environment and secondly, only the first stage could have input dependencies with other stages. This conversion is not mandatory and its purpose is to improve the performance.

The second transformation (b) is focused to channelize all the stage-in processes of a stage into a synthetic stage that is added before every stage of the workflow. Finally, if a stage is executed on the Cloud it is necessary a new transformation (c) that adds an stage for deploying the infrastructure before the stage created by (b) and another stage for the undeployment of the infrastructure after the stage-in of the subsequent stages.

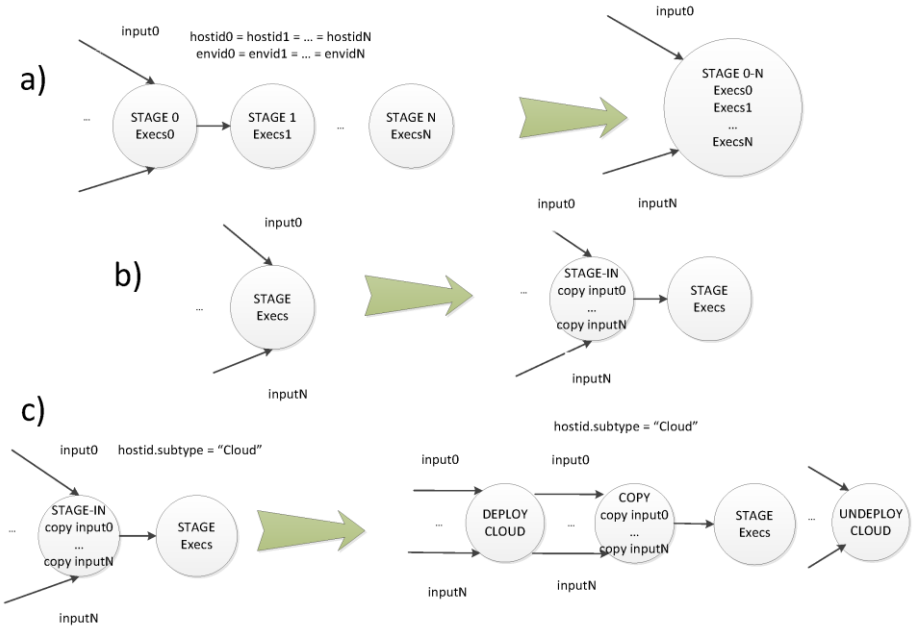


**Fig. 2.** Planner conversions

### 4.4 Runtime

The workflow execution engine is the core of the system and it is basically a runtime program in charge of monitoring the execution of the stages of the workflow and when it is necessary performing the proper actions. At the start of the execution, every stage has by default set its status to 'DISABLED'. Because the runtime is data-flow oriented, only when all the inputs of a stage are available, the status of the stage changes to 'ENABLED' and is ready to be executed. Then, the runtime periodically queries the status of the stage until it finishes. If it has finished successfully, the stage-outs (which in turn are stage-ins of subsequent stages) are enabled. The process is repeated until all the stages have been executed correctly and the final output is served to the user.

### 4.5 Connectors

The connectors are plug-ins that allow using different computational back-ends. In the Java programming language it has been implemented as an abstract class with two abstract methods, 'run_stage' and 'get_status'. Thus, when the runtime launches a stage or queries the status of an execution, it invokes these methods using the right connector. Currently, there are two connectors implemented: a PBS connector and a Cloud connector.

**PBS Connector** The PBS Connector is intended for executing stages on systems with a PBS (Portable Batch System) job scheduler. The task of PBS is to allocate computational tasks, i.e., batch jobs, among the available computing resources.

**IM(Cloud) Connector** The Cloud Connector uses a virtual infrastructure manager developed in the GRyCAP (Grid and High-Performance Computing Group). This manager called Infrastructure Manager [16] (on advance IM) has the following components: two APIs (XML-RPC and REST) for calling the services provided by it and cloud connectors for deploying, monitoring,etc. the Virtual Machines in different types of Cloud providers (OpenStack [17],Amazon EC2 [18],OpenNebula,etc). The IM selects the rightmost VMI (Virtual Machine Image) from a Virtual Machine Repository Catalog (VMRC) [19] and contextualizes it with another tool called Ansible [20].
One important detail when dealing with huge amounts of data to be processed in the Cloud Computing is that it is more efficient to move the computation to the data rather than the other way around. This requires having computational resources closely coupled to the servers holding the data. Cloud computing offers the chance to do this if the cloud is internally engineered with fast networking between the storage and compute servers.

### 4.6 Persistence

Due to the high-computational cost of e-Science experiments it is mandatory to allow users to interrupt the client execution and to resume it later. The persistence in the workflow system has been implemented using a No-SQL or document oriented database, called MongoDB. This database is suitable to be used by our tool because it stores the information as JSON-style documents, allowing the straightforward translation between workflow objects and database documents.

## 5 Use case: Orthosearch

This section briefly describes the use case that will exploit the system presented above. Among several topics in comparative genomics, homology inference aids on providing a better insight on species evolutionary history [21]. Orthology, one of the several scenarios related to homology, refers to genes or proteins that share a common ancestral, usually inter-species; such genes or proteins usually perform

equivalent functions in both species, although this is not mandatory.
OrthoSearch [22] is a scientific workflow, implemented as a comparative genomics pipeline designed for homology inference among Protozoa organisms. It uses a reciprocal best hits, HMM-profile based approach, having as input data both an ortholog database and an organism proteome.
Such data is processed in several steps with the aid of bioinformatics tools. Briefly, all ortholog groups are aligned with Mafft [23], generating multiple sequence alignments (MSAs); later, HMMER3 [24] [25] processes such output data, builds individual HMM profiles (*hmmbuild*) and a concatenated single file containing all the profiles (*hmmpress*) is generated. These concatenated profiles, will be confronted against the organism proteome (via *hmmsearch* and *hmmscan*).
At last, HMMER3s *hmmsearch* and *hmmscan* output data are processed in a bidirectional approach by a reciprocal best hits algorithm, which infers orthologous groups for this scenario (orthologous database versus organism proteome). As more organisms proteomes are submitted to OrthoSearch against the same orthologous database, common orthologous groups might be identified and analyzed in order to provide evolutionary data insight on such organisms.
Mafft, as well as HMMER3s *hmmbuild*, receives multiple files as input data and generates multiple output files; HMMER3s hmmsearch confronts each of the HMM profiles, one a at time, against the organism proteome, generating one file for each HMM profile input data; and HHMER3s hmmscan confronts a single, binary compressed data collection of profiles (generated by *hmmpress*) against the organism proteome, generating a single output file (see Figure 3).

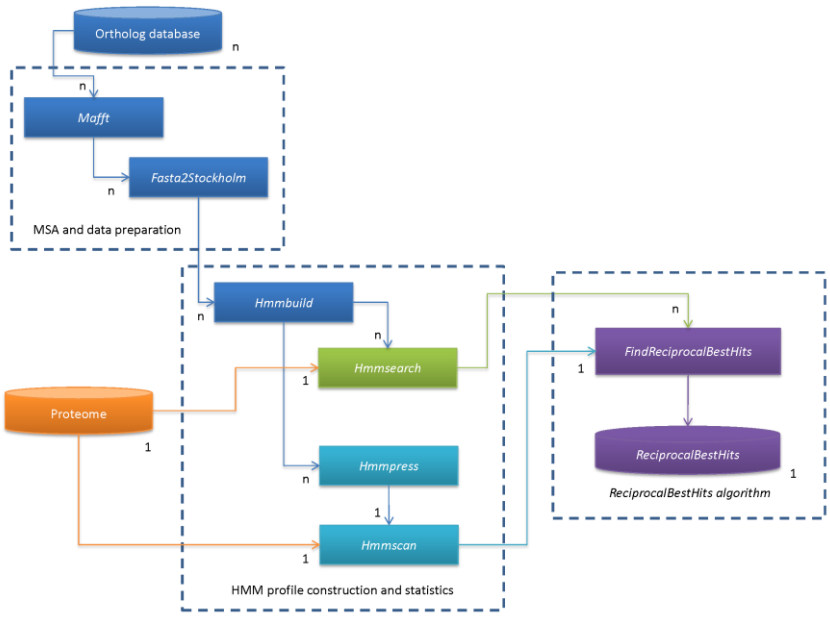


**Fig. 3.** Orthosearch workflow

Both Mafft and HMMER3 are applications which demand extensive computational power and that account for considerate amount of execution time. OrthoSearch effectiveness has been demonstrated while confronting five Protozoa species against COG/KOG databases [26] in order to infer common orthologous groups [27].
At it was mentioned before, at this moment, the entry point to the workflow system is the JSON document, and the only effort needed to analyse the Orthosearch use case in the platform is to define a simple JSON document. In the future, when the GUI is available, this task will be more automatic.

## 6 Conclusions and Future Work

The system presented in this paper is a work in progress. Although an initial prototype is available for evaluation, it will undergo significant extensions based on community feedback. The current implementation supports PBS-based systems and Cloud platforms; but efforts are being made to support other environments such as Grid. Workflow management also brings new challenges on issues like security, as it requires more flexible cooperation among different platforms. Another important aspect is that papers often draw conclusions from data that is not available to others to examine and analyse themselves, and so reproducibility is a key aspect on e-Science. These will be addressed when the tool becomes mature.
Over the long term, we believe that a system such as the one presented here can have an impact on use cases like Orthosearch by making various distributed computing infrastructures accessible to the entire science and community.

## Acknowledgements

The authors would like to thank the Spanish "Ministerio de Economía y Competitividad" for the project TIN2013-44390-R. This work has been developed under the support of the programme "Formación de Personal Investigador (FPI)" (year 2012) from the "Universidad Politécnica de Valencia", granted to Abel Carrión Collado.

## References

1. C. Hoffa, G. Mehta, T. Freeman, E. Deelman, K. Keahey, B. Berriman, and J. Good. On the Use of Cloud Computing for Scientific Workflows. In *eScience, 2008. eScience '08. IEEE Fourth International Conference on*, pages 640–645, Washington, DC, USA, December 2008. IEEE.
2. Renato J. Figueiredo, Peter A. Dinda, and José A. B. Fortes. A Case For Grid Computing On Virtual Machines. In *ICDCS '03: Proceedings of the 23rd International Conference on Distributed Computing Systems*, Washington, DC, USA, 2003. IEEE Computer Society.
3. Wei Huang, Jiuxing Liu, Bulent Abali, and Dhabaleswar K. Panda. A case for high performance computing with virtual machines. In *Proceedings of the 20th annual international conference on Supercomputing*, ICS '06, pages 125–134, New York, NY, USA, 2006. ACM.

4. Gideon Juve, Ewa Deelman, Karan Vahi, Gaurang Mehta, Bruce Berriman, Benjamin P. Berman, and Phil Maechling. Scientific workflow applications on Amazon EC2. In *2009 5th IEEE International Conference On E-Science Workshops*, pages 59–66. IEEE, December 2009.
5. T. Fahringer, R. Prodan, Rubing Duan, F. Nerieri, S. Podlipnig, Jun Qin, M. Siddiqui, Hong L. Truong, A. Villazon, and M. Wieczorek. ASKALON: A Grid Application Development and Computing Environment. In *GRID '05: Proceedings of the 6th IEEE/ACM International Workshop on Grid Computing*, GRID '05, pages 122–131, Washington, DC, USA, 2005. IEEE Computer Society.
6. *GridAnt: a client-controllable grid workflow system*, 2004.
7. *GridFlow: workflow management for grid computing*, 2003.
8. Scientific workflow management and the Kepler system. *Concurrency Computat.: Pract. Exper.*, 18:1039–1065, 2006.
9. Ewa Deelman, Gurmeet Singh, Mei H. Su, James Blythe, Yolanda Gil, Carl Kesselman, Gaurang Mehta, Karan Vahi, G. Bruce Berriman, John Good, Anastasia Laity, Joseph C. Jacob, and Daniel S. Katz. Pegasus: A framework for mapping complex scientific workflows onto distributed systems. *Scientific Programming*, 13(3):219–237, July 2005.
10. Ian Taylor, Matthew Shields, Ian Wang, and Andrew Harrison. Visual Grid Workflow in Triana. *Journal of Grid Computing*, 3(3):153–169, September 2005.
11. Tom Oinn, Matthew Addis, Justin Ferris, Darren Marvin, Martin Senger, Mark Greenwood, Tim Carver, Kevin Glover, Matthew R. Pocock, Anil Wipat, and Peter Li. Taverna: a tool for the composition and enactment of bioinformatics workflows. *Bioinformatics*, 20:3045–3054, 2004.
12. S. Woodman H. Hiden, P. Watson and D. Leahy. e-Science Central: Cloud-based e-Science and its application to chemical property modelling. 2011.
13. Jeremy Goecks, Anton Nekrutenko, James Taylor, and The Galaxy Team. Galaxy: a comprehensive approach for supporting accessible, reproducible, and transparent computational research in the life sciences. *Genome Biology*, 11:R86+, 2010.
14. M. Aleksi Kallio, Jarno Tuimala, Taavi Hupponen, Petri Klemela, Massimiliano Gentile, Ilari Scheinin, Mikko Koski, Janne Kaki, and Eija Korpelainen. Chipster: user-friendly analysis software for microarray and other high-throughput data. *BMC Genomics*, 12:507+, 2011.
15. Jia Yu and Rajkumar Buyya. A Taxonomy of Workflow Management Systems for Grid Computing, April 2005.
16. Miguel Caballer, Ignacio Blanquer, Germán Moltó, and Carlos de Alfonso. Dynamic management of virtual infrastructures. *Journal of Grid Computing*, 2014.
17. Antonio Corradi, Mario Fanelli, and Luca Foschini. VM consolidation: A real case based on OpenStack Cloud. *Future Generation Computer Systems*, June 2012.
18. F. Miller, A. Vandome, and J. McBrewster. Amazon Web Services. 2010.
19. Jose V. Carrión, Germán Moltó, Carlos De Alfonso, Miguel Caballer, and Vicente Hernández. A Generic Catalog and Repository Service for Virtual Machine Images. In *2nd International ICST Conference on Cloud Computing (CloudComp 2010)*, 2010.
20. ANSIBLE. `http://www.ansible.com/`, 2014. [Online; accessed 14-July-2014].
21. Eugene V. Koonin. Orthologs, paralogs, and evolutionary genomics1. *Annual Review of Genetics*, 39(1):309–338, 2005.
22. Sergio M. Cruz, Vanessa Batista, Edno Silva, Frederico Tosta, Clarissa Vilela, Rafael Cuadrat, Diogo Tschoeke, Alberto M. R. Davila, Maria L. Campos, and Marta Mattoso. Detecting distant homologies on protozoans metabolic pathways using scientific workflows. *International Journal of Data Mining and Bioinformatics*, 4(3):256+, 2010.

23. Kazutaka Katoh, Kei-ichi Kuma, Hiroyuki Toh, and Takashi Miyata. MAFFT version 5: improvement in accuracy of multiple sequence alignment. *Nucleic Acids Research*, 33(2):511–518, January 2005.
24. Robert D. Finn, Jody Clements, and Sean R. Eddy. HMMER web server: interactive sequence similarity searching. *Nucleic Acids Research*, 39(suppl 2):W29–W37, July 2011.
25. Sean R. Eddy. A new generation of homology search tools based on probabilistic inference. *Genome informatics. International Conference on Genome Informatics*, 23(1):205–211, October 2009.
26. Roman L. Tatusov, Natalie D. Fedorova, John D. Jackson, Aviva R. Jacobs, Boris Kiryutin, Eugene V. Koonin, Dmitri M. Krylov, Raja Mazumder, Sergei L. Mekhedov, Anastasia N. Nikolskaya, B. Sridhar Rao, Sergei Smirnov, Alexander V. Sverdlov, Sona Vasudevan, Yuri I. Wolf, Jodie J. Yin, and Darren A. Natale. The COG database: an updated version includes eukaryotes. *BMC bioinformatics*, 4(1):41+, September 2003.
27. R. Cuadrat, S. Cruz, D. Tschoeke, E. Silva, F. Tosta, H. Juca, M. Campos, M. Mattoso, and A. Davila. An orthology-based analysis of pathogenic protozoa impacting global health: an improved comparative genomics approach with prokaryotes and model eukaryote orthologsorthologs, paralogs, and evolutionary genomics. *OMICS: A Journal of Integrative Biology*, 20143001in press.

## 8.2 Anexo B - Artigo 2 – “elastic-OrthoSearch: a cloud-based comparative genomics workflow”

**Kotowski N**, Jardim R, Davila AMR. **elastic-OrthoSearch: a cloud-based comparative genomics workflow.**

**Este artigo foi submetido para a revista *Evolutionary Bioinformatics* e está em revisão.**

***elastic-OrthoSearch: a cloud-based comparative genomics workflow***

Revista: *Evolutionary Bioinformatics*
Fator de impacto: 1,452
Tipo de manuscrito: *Short Report*
Data de submissão: 17 de Maio de 2015
Autores: Kotowski, Nelson; Jardim, Rodrigo; Dávila, Alberto.

Este artigo está diretamente associado aos objetivos específicos 2.2.1, 2.2.2, e 2.2.3. É a parte principal desta tese e apresenta a solução elastic-OrthoSearch, um *workflow* científico de genômica comparativa que executa em ambiente computacional distribuído – a nuvem.

# elastic-OrthoSearch: a cloud-based comparative genomics workflow

994

Nelson Kotowski[1], Abel Carrión[2], Miguel Caballer[2], Ignacio Blanquer[2,3], Rodrigo Jardim[1], Alberto M.R. Dávila[1,*]

[1] Computational and Systems Biology Laboratory, Oswaldo Cruz Institute, Rio de Janeiro, RJ, Brazil, 21040-360

[2]Instituto de Instrumentación para Imagen Molecular (I3M), CSIC, Universidad Politècnica de València, Valencia, 46022, Spain

[3] GIBI 230, Research group in Biomedical Imaging, Polytechnic University Hospital La Fe, Valencia, Spain

* Corresponding author. Mailing address: Laboratório de Biologia Computacional e Sistemas, FIOCRUZ/IOC, Av. Brasil, 4365, Manguinhos, CEP 21040-360, Rio de Janeiro, RJ, Brazil. Phone (+55 21) 2562-1025. E-mail: davila@fiocruz.br

Abstract

Homology inference applications play an important role on inferring function to newly sequenced genes. OrthoSearch, a protein-profile, reciprocal best hits-based comparative genomics pipeline, aims at inferring ortholog groups among organisms. In order to address the increasing computational power needed to compute the ever-increasing amount of genomics data in an adequate timeline, we present elastic-OrthoSearch. A cloud-based update of OrthoSearch was developed, supported by a cloud-enabled workflow engine on an “on premise” OpenNebula Cloud. Supporting documentation available at: http://biowebdb.org/elastic_OrthoSearch.

## Introduction

Comparative genomics, which mainly relates to homology and evolutionary dynamics, aims at providing means for researchers to better comprehend how species evolved, through complete genome or specific genes analysis (Hardison, 2003). Homologous proteins share a common ancestral, either in the same species or among species. Although there are several concepts associated to homology -

orthology, paralogy, horizontal gene transfer (HGT), gene loss and others (Koonin, 2005) – this article focuses only on orthology and paralogy aspects.

Genes or proteins might be inferred as orthologs when such are present in different species, due to a speciation event. At the same time, paralogs are usually duplicated genes in the same species, although they may occur in distinct ones (Koonin, 2005).

As orthologs tend to preserve their ancestor function, their inference poses an important issue in transferring function to recently sequenced genes. In addition, such studies aid on providing a better insight on genes evolutionary history and consequently, to the species evolution (Koonin, 2005).

Among many tools which aid on homology inference between species, OrthoSearch (da Cruz et al., 2010; da Cruz et al., 2008) is a profile-protein, reciprocal best hits (RBH) based comparative genomics pipeline. It has already proven to be effective inferring orthologous groups among Protozoa species (Cuadrat et al., 2014) with respect to COG/KOG orthologous databases (Tatusov et al., 2003).

Cloud computing is a particular distributed computing paradigm (Mell & Grance, 2011), which dynamically provides resources (computing, storage, network and higher-level services) from a multi-tenant pool. Although there is an increasing interest on computing clouds solutions, the management of the virtual infrastructure, data transferring, and lack of adequate software prevent users from moving to such environment.

One recent comparative genomics algorithm which has been adapted to the cloud is the Reciprocal Smallest Algorithm (RSD) (DeLuca et al., 2012; Wall et al., 2010). Also, CloudBLAST (Matsunaga et al., 2008) and Crossbow (Gurtowski et al., 2012) provide other good examples on how comparative genomics might benefit from such computing paradigm.

This work uses a workflow engine (Carrión et al., 2014) that eases the instantiation of any workflow that can be expressed as a DAG (Directed Acyclic Graph). Although the tool offers the possibility of using a mix of computing back-ends, elastic-OrthoSearch was executed on the cloud. The benefits offered by the engine are twofold: firstly, there is no need for ad-hoc hardware to obtain good performance results; secondly, in contrast to other workflow engine solutions, the one

used in this work is optimized to exploit key features of the cloud, such as the dynamic provisioning of cloud computing resources.

The engine leverages from the Infrastructure Manager (Caballer et al., 2015) (Caballer et al., 2015) and the Virtual Machine Image Repository & Catalog (Carrión et al., 2010) components for automating selection, deployment, configuration, software installation, monitoring and update of VMIs (Virtual Machine Image).

## Material and Methods

Both OrthoSearch and elastic-OrthoSearch require an orthologous database and an organism multifasta protein data as input data. Due to hardware limitations, which could pose many difficulties to processing the largest orthologous group files we selected a subset of Eggnog version 4 (Powell et al., 2013), comprising eukaryotic clusters (euNOG members). We did such by allowing only up to two random proteins for each of the 282 organisms to be present at each of the 60,164 orthologous groups.

The organisms confronted against such database were: *Cryptosporidium hominis*, *Entamoeba histolytica* and *Leishmania infantum*. They were randomly selected and such protein data was obtained via GenBank.

We started by executing the non-elastic OrthoSearch application in a single Ubuntu 14.04 compute instance with 16 CPU cores, 24GB RAM and 100GB disk. Later, elastic-OrthoSearch was executed on an on premise cloud infrastructure managed by OpenNebula 4.8 cloud middleware, with up to 16 Ubuntu 14.04 contextualized VMs.

All experiments were performed for each of the 3 Protozoa organisms against the same orthologous database in every computing scenario, in a total of 15 experiments.

There are five designed stages for elastic-OrthoSearch, comprising “mafft-fasta2stockholm-hmmbuild”; “hmmsearch”; “cat”; “hmmpress-hmmscan” and “best-hits”.

We provide elasticity to our pipeline by allowing the user to specify different hardware and software specifications for each of such stages previously to the pipeline execution (Supplementary Data 1).

All of these configurations are managed by the Infrastructure Manager (IM) (Caballer et al., 2015), which is in charge of deploying the required number of VM instances. For speedup testing, four different cloud scenarios were assembled, with 2, 4, 8 and 16 parallel instances each.

## Results

Our workflow machine registers the elapsed time for each workflow stage and with that we were able to calculate the total execution time for each experiment (VM deployment, data transfer and VM deallocation included).

With such data, we inferred the time speedup ratio, a measure that shows how many times we were able to improve our execution time for each scenario against the non-elastic OrthoSearch experiments. (Table 1 approximately here).

In a 16 instances scenario with *Cryptosporidium hominis* we obtained a 4.95 time speedup ratio with a total 06 hours and 54 minutes, against 34 hours and 10 minutes in the non-elastic OrthoSearch pipeline. Our worst-case scenario, *Leishmania infantum*, using 2 instances, provided a remarkable 1.99 speedup.

(Figure 1 approximately here)

elastic-OrthoSearch has a limited scalability, shown by the stabilization of the speedup as the number of instances grown. This relates to the pipeline structure. Some of its tasks run in parallel, while others require the completion of previous stages. Such could keep on waiting state while, for example, Mafft performs an extremely large orthologous group alignment. In addition, some stages are not eligible to parallelization, as CAT and HMMER3 “hmmpress/hmmscan”.

## Conclusions

This study presented elastic-OrthoSearch, a cloud computing based comparative genomics pipeline, supported by a DAG-enabled workflow engine. We obtained promising results, with significant speedup when compared to the original, batch-oriented pipeline.

Although there are known limitations related to achieving an ideal number of instances, the obtained performance is encouraging. The costs of smaller but multiple instances at the cloud compensate for the required investment on clusters or supercomputers and maintenance for such on-demand experiments. Moreover, sequential parts may be substituted by parallel-computing legacy versions of the components.

elastic-OrthoSearch might enhance future homology inference experiments by providing results in an improved, shorter timeline, without requiring up-front investments on a computing infrastructure, what could empower scientists in comparative genomics analysis.


### Acknowledgements

The authors thank CAPES Foundation (http://capes.gov.br) and the Spanish "Ministerio de Economía y Competitividad", for the project TIN2013-44390-R "Clusters Virtuales Elásticos y Migrables sobre Infraestructuras Cloud Híbridas".

### Funding sources

This work was supported by the CAPES Foundation grant BEX 2137/14-3 to NK and the Universitat Politècnica de València for the FPI 2012 grant to AC.


### Author Contributions

Conceived and designed the experiments: NK, AC, RJ, MC. Analysed the data: NK, AC, MC, RJ. Wrote the manuscript first draft: NK, AC. Contributed to the writing of the manuscript: NK, AC, MC, IB, RJ, AMRD. Agreed with manuscript results and conclusions: NK, AC, MC, IB, RJ, AMRD. Made critical revisions and approved final version: AMRD, IB. All authors reviewed and approved of the final manuscript.

### Competing Interests

The authors declare that there are no competing interests.

### Disclosures and Ethics

As a requirement of publication author(s) have provided to the publisher signed confirmation of compliance with legal and ethical obligations including but not limited to the following: authorship and contributorship, conflicts of interest, privacy and confidentiality and (where applicable) protection of human and animal research subjects. The authors have read and confirmed their agreement with the ICMJE authorship and conflict of interest criteria. The authors have also confirmed that this article is unique and not under consideration or published in any other publication, and that they have permission from rights holders to reproduce any copyrighted material. Any disclosures are made in this section. The external blind peer reviewers report no conflicts of interest.

## References

Caballer,M. et al. (2015) Dynamic Management of Virtual Infrastructures. J. Grid Comput., 13, 53–70.

Carrión,A. et al. (2014) Design and implementation of a Generic and Multi-Platform Workflow System. In, 8th Iberian Grid Infrastructure Conference Pro-ceedings., p. 77.

Carrión,J.V. et al. (2010) A Generic Catalog and Repository Service for Virtual Machine Images.

Da Cruz,S.M.S. et al. (2010) Detecting distant homologies on protozoans metabolic pathways using scientific workflows. Int. J. Data Min. Bioinforma., 4, 256–280.

Da Cruz,S.M.S. et al. (2008) OrthoSearch: a scientific workflow approach to detect distant homologies on protozoans. In, SAC ’08. ACM, New York, NY, USA, pp. 1282–1286.

Cuadrat,R.R.C. et al. (2014) An Orthology-Based Analysis of Pathogenic Protozoa Impacting Global Health: An Improved Comparative Genomics Ap-proach with Prokaryotes and Model Eukaryote Orthologs. OMICS J. In-tegr. Biol., 140624130015005.

DeLuca,T.F. et al. (2012) Roundup 2.0: Enabling comparative genomics for over 1800 genomes. Bioinformatics.

Gurtowski,J. et al. (2012) Genotyping in the Cloud with Crossbow. In, Baxevan-is,A.D. et al. (eds), Current Protocols in Bioinformatics. John Wiley & Sons, Inc., Hoboken, NJ, USA.

Hardison,R.C. (2003) Comparative Genomics. PLoS Biol., 1, 156–160.

Koonin,E.V. (2005) Orthologs, Paralogs, and Evolutionary Genomics. Annu. Rev. Genet., 39, 309–338.

Matsunaga,A. et al. (2008) CloudBLAST: Combining MapReduce and Virtualization on Distributed Resources for Bioinformatics Applications. IEEE, pp. 222–229.

Mell,P. and Grance,T. (2011) The NIST Definition of Cloud Computing.

Powell,S. et al. (2013) eggNOG v4.0: nested orthology inference across 3686 organ-isms. Nucleic Acids Res., gkt1253.

Tatusov,R.L. et al. (2003) The COG database: an updated version includes eukary-otes. BMC Bioinformatics, 4, 41.

Wall,D.P. et al. (2010) Cloud computing for comparative genomics. BMC Bioinfor-matics, 11, 259.

## Tables

Table 1

| | Execution time (HH:MM) | | |
|---|---|---|---|
| Instances | *Cryptosporidium hominis* | *Entamoeba histolytica* | *Leishmania infantum* |
| 1 | 34:10 | 39:07 | 39:34 |
| 2 | 16:36 | 19:35 | 19:52 |
| 4 | 10:43 | 14:06 | 14:27 |
| 8 | 08:20 | 11:31 | 10:52 |
| 16 | 06:54 | 10:11 | 09:35 |

## Figures

Figure 1

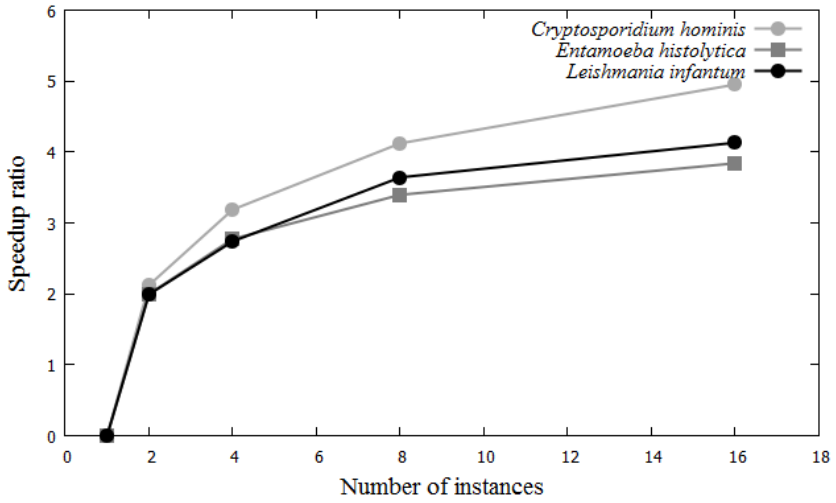


Figure 1: Speedup ratio obtained by elastic-OrthoSearch according to each Protozoa organism and the number of instances at the cloud.

## Supplementary data

Supplementary Data 1.json

```
{
"hosts":[
     {
          "hostId": "one1",
          "type": "Cloud",
          "subType": "OpenNebula",
          "hostName": "cloud_front_end",
```

```
            "port": "2633",
            "credentials":{
                "userName": "user",
                "password": "password"
            }
        }

    ],
    "environments": [
        {
            "environmentId": "ubuntu64bit",
            "osName": "linux",
            "arch": "x86_64",
            "osFlavour": "ubuntu",
            "osVersion": "14.04",
            "packages":[
                "MAFFT",
                "HMMER",
                "unzip",
                "dos2unix",
                "ruby"
            ]
        }
    ],
    "stages":[
        {
            "id": "mafft-fasta2stockholm-hmmbuild",
            "hostId": "#one1",
            "environmentId": "#ubuntu64bit",

            "nodes": [
                {
                        "numNodes":"4",
                        "coresPerNode":"1",
                        "memorySize":"2048m",
                        "disks": [
```

```
                {
                    "nDisk": "0",
                        "diskSize":"40g"
                    }
                ]
        }
    ],

    "execution": [
        {
            "path": "./mafft_st_hmmbuild",
            "arguments": "#input2(1)"
        }
    ],

    "retries" : {
        "onWallTimeExceeded": "0",
        "onSoftwareFailure": "0",
        "onHardwareFailure": "0"
    },

    "stageIn": [
        {
            "id": "input0",
                "type": "File",
            "values": [
                "mafft_st_hmmbuild"
            ]
        },
        {
            "id": "input1",
                "type": "File",
            "values": [
                "fasta2stockholm.pl"
            ]
        },
```

```
            {
                "id": "input2",
                    "type": "File",
                "values": [
                    "eggnog_kog.zip(extract)"
                ]
            }
        ],

        "stageOut": [
            {
                    "id": "output0",
                    "type": "File",
                    "filterIn": "*.mafft.st.hmm",
                    "replica": "None"
            }
        ]
    },
    {
        "id": "hmmsearch",
        "hostId": "#one1",
        "environmentId": "#ubuntu64bit",

             "nodes": [
            {
                    "numNodes":"4",
                    "coresPerNode":"1",
                    "memorySize":"4096m",
                    "disks": [
                    {
                        "nDisk": "0",
                            "diskSize":"40g"
                        }
                    ]
            }
        ],
```

```
                "execution": [
                     {
                          "path": "hmmsearch",
                          "arguments":   "-E   0.1   #output0(1)
#input3 > #output0(1).#input3"
                     }
                ],

                "retries" : {
                     "onWallTimeExceeded": "0",
                     "onSoftwareFailure": "0",
                     "onHardwareFailure": "0"
                },

                "stageIn": [
                     {
                          "id": "#output0"
                     },
                     {
                          "id": "input3",
                               "type": "File",
                          "values": [
                               "Cryptosporidium-hominis-TU502-
cdhit-s1.fasta"
                          ]
                     }
                ],

                "stageOut": [
                     {
                               "id": "output1",
                               "type": "File",
                               "filterIn": "*.mafft.st.hmm.*",
                               "replica": "None"
                     }
```

```
        ]
    },
    {
        "id": "cat",
        "hostId": "#one1",
        "environmentId": "#ubuntu64bit",

            "nodes": [
            {
                    "numNodes":"1",
                    "coresPerNode":"1",
                    "memorySize":"4096m",
                    "disks": [
                    {
                        "nDisk": "0",
                            "diskSize":"40g"
                        }
                    ]
            }
        ],

        "execution": [
            {
                "path": "cd",
                "arguments": "output0"
            },
            {
                "path": "ls",
                "arguments": ". | xargs -n 32 cat >
$JOBID/#output2"
            }
        ],

        "retries" : {
            "onWallTimeExceeded": "0",
            "onSoftwareFailure": "0",
```

```
            "onHardwareFailure": "0"
        },

        "stageIn": [
            {
                "id": "#output0"
            }
        ],

        "stageOut": [
            {
                    "id": "output2",
                    "type": "File",
                    "replica": "None",
                "values": [
                    "hmmcat.hmm"
                ]
            }
        ]
    },
    {
        "id": "hmmerpress-hmmerscan",
        "hostId": "#one1",
        "environmentId": "#ubuntu64bit",

             "nodes": [
            {
                    "numNodes":"1",
                    "coresPerNode":"1",
                    "memorySize":"4096m",
                    "disks": [
                    {
                        "nDisk": "0",
                            "diskSize":"40g"
                        }
                    ]
```

```
                    }
               ],

               "execution": [
                    {
                         "path": "hmmpress",
                         "arguments": "#output2"
                    },
                    {
                         "path": "hmmscan",
                         "arguments":   "#output2   #input4   >
#input4.out"
                    }
               ],

               "retries" : {
                    "onWallTimeExceeded": "0",
                    "onSoftwareFailure": "0",
                    "onHardwareFailure": "0"
               },

               "stageIn": [
                    {
                         "id": "#output2"
                    },
                    {
                         "id": "input4",
                              "type": "File",
                         "values": [
                              "Cryptosporidium-hominis-TU502-
cdhit-s1.fasta"
                         ]
                    }
               ],

               "stageOut": [
```

```
                    {
                              "id": "output3",
                              "type": "File",
                         "filterIn": "*.out",
                              "replica": "None"
                    }
               ]
          },
          {
               "id": "best-hits",
               "hostId": "#one1",
               "environmentId": "#ubuntu64bit",

                     "nodes": [
                    {
                              "numNodes":"1",
                              "coresPerNode":"1",
                              "memorySize":"4096m",
                              "disks": [
                              {
                                   "nDisk": "0",
                                        "diskSize":"40g"
                                   }
                              ]
                    }
               ],

               "execution": [
                    {
                         "path": "ruby",
                         "arguments":  "Run.rb  -O  output1  -H
#output3 -C ko_vs_chominis.csv"
                    }
               ],

               "retries" : {
```

```
                "onWallTimeExceeded": "0",
                "onSoftwareFailure": "0",
                "onHardwareFailure": "0"
            },

            "stageIn": [
                {
                    "id": "#output1"
                },
                {
                    "id": "#output3"
                },
                {
                    "id": "input5",
                        "type": "File",
                    "values": [
                        "besthits_src.zip(extract)"
                    ]
                }
            ],

            "stageOut": [
                {
                        "id": "output4",
                        "type": "File",
                    "filterIn": "*.csv",
                        "replica": "None"
                }
            ]
        }
    ]
    }
```

## 8.3 Anexo C - Artigo 3 - “Improved orthologous databases to ease Protozoan targets inference”

**Kotowski N**, Jardim R, Davila AMR. **Improved orthologous databases to ease Protozoan targets inference.**

**Este artigo foi publicado na revista *Parasites & Vectors* em 29 de Setembro de 2015.**

***Improved orthologous databases to ease Protozoan targets inference***

Revista: *Parasites&Vectors*

Fator de Impacto: 3,43

Tipo de manuscrito: *Research*

Data de submissão: 02 de Junho de 2015

Data de publicação: 29 de Setembro de 2015

Autores: Kotowski, Nelson, FIOCRUZ; Jardim, Rodrigo; FIOCRUZ, Dávila, Alberto M R; FIOCRUZ

doi: 10.1186/s13071-015-1090-0

Associado ao objetivo específico 2.2.4, este artigo apresenta a criação de uma metodologia baseada no OrthoSearch que possibilitou a construção de bases de dados de grupos ortólogos. Estas bases possuem o potencial de alavancar a inferência de homologia entre organismos, a transferência de anotações e a identificação de alvos em protozoários, entre outras possibilidades.

Adicionalmente, é neste estudo que apresentamos uma versão atualizada do OrthoSearch, na qual adotamos a versão 3 do pacote *HMMER* e as linguagens C e *Ruby* em detrimento da dependência de sistema de gerenciamento de *workflows* científicos.

Este artigo relaciona-se com esta tese, pois foi esta a versão do OrthoSearch utilizada como base durante todo o estudo e por possibilitar a criação de bases que podem ser futuramente utilizadas em experimentos com o elastic-OrthoSearch.

Kotowski *et al. Parasites & Vectors* (2015) 8:494
DOI 10.1186/s13071-015-1090-0

Parasites & Vectors

RESEARCH Open Access

# Improved orthologous databases to ease protozoan targets inference

Nelson Kotowski, Rodrigo Jardim and Alberto M. R. Dávila*

## Abstract

**Background:** Homology inference helps on identifying similarities, as well as differences among organisms, which provides a better insight on how closely related one might be to another. In addition, comparative genomics pipelines are widely adopted tools designed using different bioinformatics applications and algorithms. In this article, we propose a methodology to build improved orthologous databases with the potential to aid on protozoan target identification, one of the many tasks which benefit from comparative genomics tools.

**Methods:** Our analyses are based on OrthoSearch, a comparative genomics pipeline originally designed to infer orthologs through protein-profile comparison, supported by an HMM, reciprocal best hits based approach. Our methodology allows OrthoSearch to confront two orthologous databases and to generate an improved new one. Such can be later used to infer potential protozoan targets through a similarity analysis against the human genome.

**Results:** The protein sequences of *Cryptosporidium hominis*, *Entamoeba histolytica* and *Leishmania infantum* genomes were comparatively analyzed against three orthologous databases: (i) EggNOG KOG, (ii) ProtozoaDB and (iii) Kegg Orthology (KO). That allowed us to create two new orthologous databases, "KO + EggNOG KOG" and "KO + EggNOG KOG + ProtozoaDB", with 16,938 and 27,701 orthologous groups, respectively.
Such new orthologous databases were used for a regular OrthoSearch run. By confronting "KO + EggNOG KOG" and "KO + EggNOG KOG + ProtozoaDB" databases and protozoan species we were able to detect the following total of orthologous groups and coverage (relation between the inferred orthologous groups and the species total number of proteins): *Cryptosporidium hominis*: 1,821 (11 %) and 3,254 (12 %); *Entamoeba histolytica*: 2,245 (13 %) and 5,305 (19 %); *Leishmania infantum*: 2,702 (16 %) and 4,760 (17 %).
Using our HMM-based methodology and the largest created orthologous database, it was possible to infer 13 orthologous groups which represent potential protozoan targets; these were found because of our distant homology approach.
We also provide the number of species-specific, pair-to-pair and core groups from such analyses, depicted in Venn diagrams.

**Conclusions:** The orthologous databases generated by our HMM-based methodology provide a broader dataset, with larger amounts of orthologous groups when compared to the original databases used as input. Those may be used for several homology inference analyses, annotation tasks and protozoan targets identification.

**Keywords:** Comparative genomics, Homology inference, Target identification, Protozoa, Orthologous database, Distant homology, *Leishmania*, *Cryptosporidium*, *Entamoeba*

* Correspondence: davila@fiocruz.br
Computational and Systems Biology Laboratory, Oswaldo Cruz Institute, FIOCRUZ, Avenida Brasil, 4365, 21040-360 Rio de Janeiro, RJ, Brazil

## Background

Historically, the very definition of a Protozoa represents an open debate. Despite many classifications and changes provided over history [1–4], in this article we will refer to Protozoa as eukaryotic organisms, apart from those who do not have a primitive mitochondria, peroxisomes (*Archezoa*) and the shared characteristics which define the *Animalia*, *Fungi*, *Plantae* and *Chromista* kingdoms [4].

There are over 200,000 described protozoan organisms and among them over 10,000 parasites of invertebrate organisms and nearly all vertebrate ones [1]. There are several protozoan related diseases, which affect more than 25 % of the world population, such as Chagas' disease, Human African Trypanosomosis, Leishmaniosis, Amoebiosis, Giardiosis, Toxoplasmosis, Cryptosporidiosis, Theileriosis, Babesiosis among many others [5–10].

Neglected Tropical Diseases (NTDs) are diseases caused by a variety of organisms and are usually associated to developing countries, which suffer from poor sanitation, hygiene, social and financial conditions. Over 1 billion people are affected by such diseases, in 149 countries worldwide [11]. Among the 17 NTDs listed by WHO, three are caused by protozoan organisms: Chagas' disease (*Trypanosoma cruzi*), Human African Trypanosomosis (*Trypanosoma brucei*) and Leishmaniosis (*Leishmania* spp.) [11].

According to the 3rd WHO report on NTDs, even though several advances have been achieved in the recent years, there is a permanent need for research and innovation in improved diagnosis, next-generation treatments and interventions for such NTDs [12, 13].

Leishmaniosis is a neglected disease caused by the *Leishmania* spp. and transmitted by phlebotomine sandflies [14]. More than 1.3 million people are infected worldwide, especially those who live in poor sanitation, hygiene and social conditions and WHO estimates about 20,000 to 30,000 deaths occur yearly [14]. Such disease has three distinct presentations: cutaneous, visceral and mucocutaneous, each of them related to different *Leishmania* spp. and world regions.

Leishmaniosis is hard to diagnose and treat. So far, the available drugs and vaccines are either toxic or present poor efficiency [15]. Also, its elevated treatment cost (up to US$252/patient, depending on the applied treatment and drugs) eventually becomes prohibitive to the most affected and poor countries [15].

Besides allowing for a better comprehension of such Protozoa organism, many molecular studies have been done over the last few years using DNA/RNA sequencing methodologies [16–18], which have been used in order to infer new drug targets. These data are available in several public databases and allow for comparative genomics studies among either closely or distant related organisms. Also, that might increase the odds of discovering relevant information applied to drug manufacturing or reuse, which could be later applied to disease treatments.

Comparative genomics mainly refers to homology and evolutionary dynamics between organisms, genes and proteins, which provides better understanding on how species evolved through comparing either their complete genomes or specific genes [19]. Homologous genes share a common ancestry, either intra- or inter-species. Several scenarios relate to homology, such as orthology, paralogy, horizontal gene transfer, gene loss, orphan genes and others; for this study, we will focus on orthology aspects only [20].

Orthology might be inferred when the same genes or proteins are present in distinct species, and this was due to a speciation event [20].

Homology inference has become an important issue when inferring function to recently sequenced genes because orthologs tend to preserve their ancestor function. Besides that, such studies provide a better insight on genes evolutionary history and consequently, to the species evolution [20, 21].

Inferring putative function is one of the particular benefits in orthologous group (OG) assignment, especially when dealing with recently sequenced genome data [22]. In addition, OGs may provide us a better comprehension on species evolutionary relationships [23], since it is through such data that one might provide information that could help on both evolutionary and functional analysis [24].

Moreover, several tasks could benefit from OGs, such as genome annotation, gene conservation, protein family identification, phylogenetic tree reconstruction, pharmacology and many others [22, 24–27]. Topics as positional orthology and synteny conservation among orthologs are also appealing to those who aggregate genomic context in their homology inference methods [28].

There are several available methodologies to aid on homology detection. Besides a simple categorization effort [26], we will follow Dalquen's proposition [29]. Briefly, three distinct approaches are available: (i) the one which use multiple sequence alignment (MSA) scores along with reciprocal best hits, such as OrthoSearch [30], OrthoMCL [24] and InParanoid [31]; (ii) that which rely on evolutionary distance calculus, as RSD [32, 33]; (iii) and that based on phylogenetic trees reconstruction, as SPIMAP [34].

Many orthologous databases (OD) are created by homology inference methods. This is the case for OrthoMCLDB [35]; InParanoid [36]; Roundup [32]; COG/KOG [37] and EggNOG [38, 39].

OrthoSearch [40] is a scientific workflow [41] for homology inference among species. Initially conceived as a Perl-based routine, it uses a reciprocal best hits, HMM-based approach. OrthoSearch has already proven to be effective

inferring orthology among five protozoan genomes, using COG and KOG ODs [27].

In this work, we propose an update and a new functionality for OrthoSearch, showing it as an effective tool in providing means to create new ODs (n-ODs). So far, we tested our methodology in a controlled, three steps scenario: (i) Protozoa orthology inference and (ii) n-ODs creation, both supported by publicly available ODs used as input; and (iii) improved Protozoa orthology inference, supported by such recently created n-ODs.

With our methodology and generated n-ODs, we expect to be able to provide ODs with broader data sets, which in turn can be applied in target identification for protozoan organisms, such as stated by Timmers *et al.* [42] review on research efforts related to genomic database development for protozoan parasites.

Moreover, previous initiatives, such as the study performed by Tschoeke *et al.* [18] regarding the *Leishmania amazonensis* parasite, as well as the *Leishmania donovani* comparative genomics analysis performed by Satheesh *et al.* [43] corroborate the benefits provided by the use of broader orthologous data sets.

## Methods

### OrthoSearch improvements and analyses scenarios

In order to reach our main methodological goal, which is to provide OrthoSearch with means to create n-ODs, we revisited its original pipeline. Notably (i) we adopted HMMER version 3 and (ii) changed from a Perl-based routine to C++ 4.43 and Ruby 1.8.7 modules. A dedicated Ubuntu 12.04 single-server machine with 64 cores and 32GB RAM was used for all assembled scenarios.

### OrthoSearch for protozoa orthology inference

OrthoSearch needs as input data an (i) OD and (ii) an organism multifasta protein data. We used Kegg Orthology (KO) [44, 45], EggNOG KOG and ProtozoaDB as input ODs. KO, downloaded via FTP, contains data from all life domains – Archaea, Bacteria and Eukarya. EggNOG KOG is a eukaryotic-only groups Eggnog subset [39], downloaded directly from its website. ProtozoaDB OGs [46] (which contain only protozoan species) were also used for our analyses. Details about each OD are available in Additional file 1.

We randomly selected three protozoan species as OrthoSearch organisms input data: *Cryptosporidium hominis*, *Entamoeba histolytica* and *Leishmania infantum*, each with 3,885, 7,973 and 7,872 proteins and downloaded such data from ProtozoaDB [46].

### OrthoSearch for orthologous database building

Figure 1 depicts OrthoSearch pipeline with its two possibilities, which could be (i) a standard orthology inference or (ii) a n-OD creation. In order to build the n-ODs, we used as input data an OD in its original composition; and another OD data subset, which enacts as an organism multifasta protein data.

That was called an impersonated proteome and was generated by choosing a representative protein for each OG at the confronted OD. The selection and extraction of such proteins was performed with the support of a Python script kindly developed by Salvador Capella (personal communication, URL: https://github.com/scapella/trimal/blob/dev/scripts/get_sequence_representative_from_alignment.py) and an internally developed Ruby script. Each representative protein identifier and its amino acid sequence were stored in a single multifasta file (see Additional file 2).

We started with KO as our fixed, complete OD, against an impersonated EggNOG KOG multifasta protein data. Reciprocal best hits between both of them were processed using internal scripts developed in both Ruby and Unix/POSIX shell script languages, so that, new OGs were created and arranged, generating the n-OD conveniently named "KO + EggNOG KOG".

During such n-OD creation, there were three possible scenarios related to the fixed database OGs – KO – and the impersonated proteome. Briefly, (i) those OGs from KO database which did not have any reciprocal best hit with the impersonated OGs from EggNOG KOG; and (ii) those OGs from EggNOG KOG that did not have any reciprocal best hit against KO database were identified, selected and incorporated to our n-OD ("KO + EggNOG KOG"), without any changes; (iii) those OGs from KO database which presented a reciprocal best hit with the impersonated OGs from EggNOG KOG were expanded, by adding up every respective EggNOG KOG OG proteins into such KO group.

Therefore, at the end of this run, we had a n-OD called "KO + EggNOG KOG" which comprised original, unaltered KO and EggNOG KOG OGs, as well as expanded KO OGs, which now also contain EggNOG KOG protein data. Once such n-OD was created, we performed the same steps described above, confronting "KO + EggNOG KOG" against ProtozoaDB impersonated OGs, which generated our second n-OD called "KO + EggNOG KOG + ProtozoaDB".

### Inferring protozoan orthologs with OrthoSearch and the n-ODs

Having built the two n-ODs: "KO + EggNOG KOG" and "KO + EggNOG KOG + ProtozoaDB", we analyzed OrthoSearch in a standard orthology inference against the three above mentioned protozoan species.

### Comparison between the n-ODs and OrthoMCLDB

The three protozoan species were confronted against OrthoMCLDB through online phyletic pattern search queries, in order to infer its orthologous proteins, in the

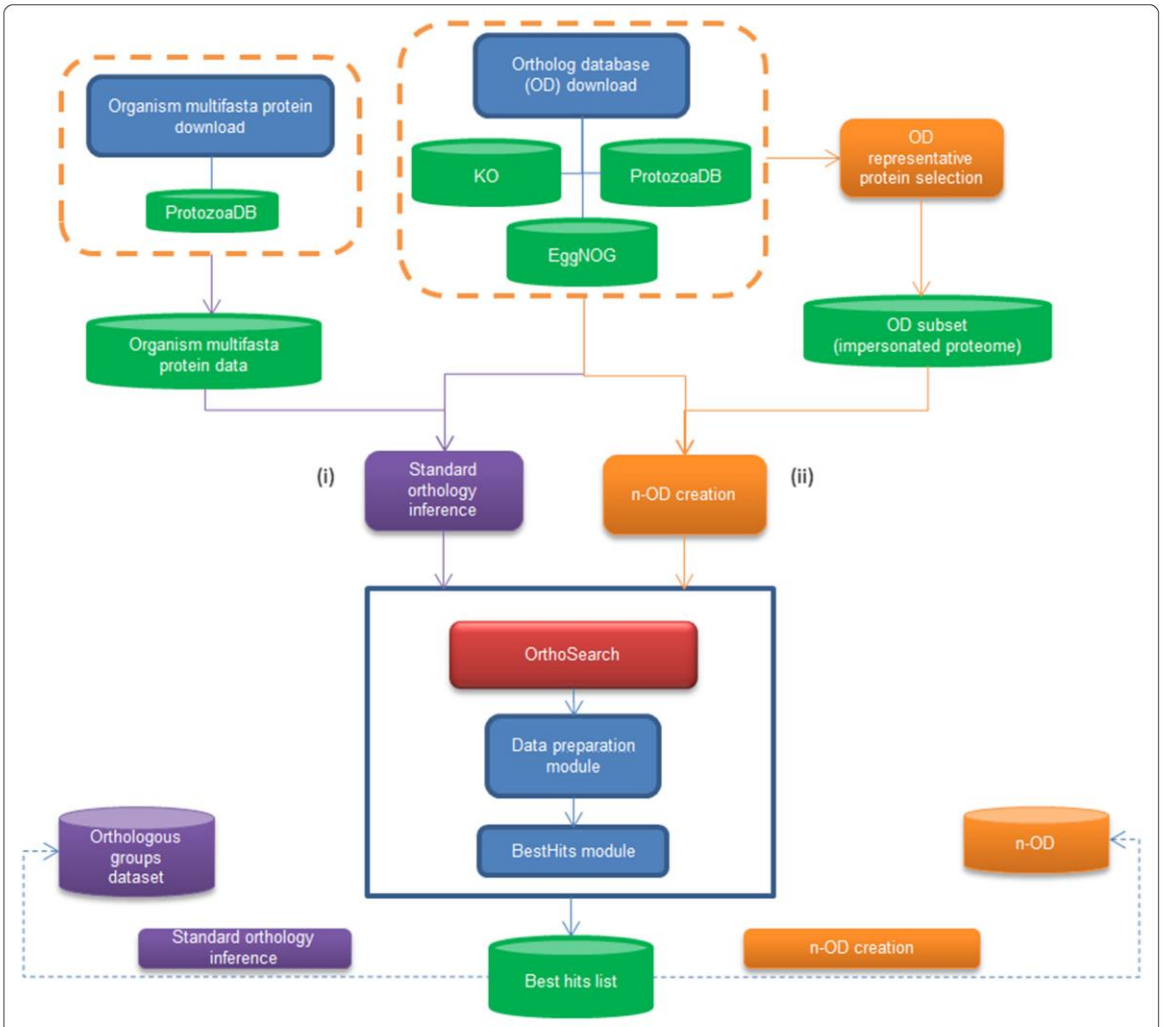


**Fig. 1** OrthoSearch pipeline and the two possible strategies; This figure describes how OrthoSearch pipeline works in a (i) standard orthology inference; and how it is reused to (ii) create a n-OD

same way as we did with the n-ODs created by our proposed methodology.

Quantitative results obtained while executing both OrthoSearch (with the two n-ODs) and OrthoMCLDB against the three protozoan species were compared in order to offer a better understanding of the proposed methodology behavior and to analyze if we were able to provide better results or not.

### Potential *Leishmania* spp. targets against the human proteome

In order to identify potential protozoan targets that are not available at the human genome, a BlastP [47] was performed between the largest n-OD generated by our methodology - "KO + Eggnog KOG + ProtozoaDB" (details on the orthologous groups proteins are available at Additional file 3) and the human proteome, downloaded via RefSeq [48]. We used BlastP 2.2.28+ with 0.1 as e-value, extracted and analyzed the orthologous groups which did not perform any hit against the human proteome but provided results against *Leishmania* spp. and therefore could represent potential targets. A BlastP was also performed against KO, Eggnog KOG and ProtozoaDB orthologous databases separately.

## Results

### OrthoSearch for protozoa orthology inference

The protein data of the three protozoan species were confronted against (i) KO, (ii) EggNOG KOG and (iii) ProtozoaDB ODs (Fig. 2). ProtozoaDB performed best,

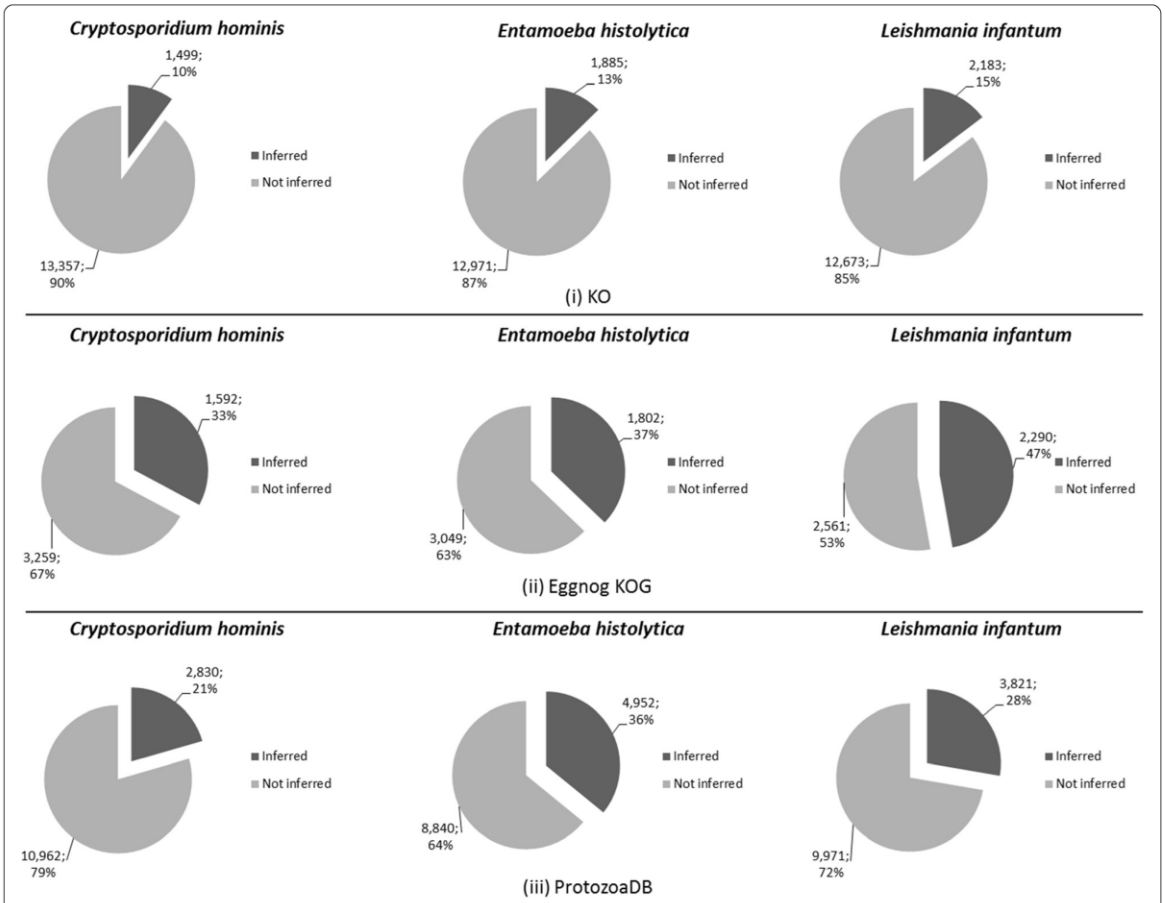


**Fig. 2** OrthoSearch inferred orthologous groups and coverage per organism; A detailed view on how many orthologous groups were inferred with (i) KO, (ii) EggNOG KOG and (iii) ProtozoaDB databases and what do such numbers represent against the organisms total protein numbers

with: 2,830 OGs against *Cryptosporidium hominis*, 4,952 for *Entamoeba histolytica* and 3,821 for *Leishmania infantum*.

With such data, we extracted coverage percentage information, which shows the total number of OGs inferred by OrthoSearch versus how many OGs are contained within each OD. For *Cryptosporidium hominis*, which has the smallest number of proteins of the three protozoan species studied, EggNOG KOG performed best, with 33 % coverage. *Entamoeba histolytica* also performed well with EggNOG KOG (37 %), but showed very similar results with ProtozoaDB (36 %), while showing a poor coverage with KO (13 %). Finally, *Leishmania infantum* had the best coverage (47 %), with EggNOG KOG.

Internal scripts, developed with the R language and its Venn Diagram library, processed reciprocal best hits for such protozoan species. We identified species-specific, pair-to-pair and core OGs, depicted at Fig. 3.

EggNOG KOG presented the best core (ratio between OGs shared by the three protozoan species and all the inferred OGs), corresponding to 26.72 % of the total inferred best hits (828/3,099 OGs), followed by KO - 20.94 % (719/3,434 OGs) and ProtozoaDB - 4.65 % (464/9,979 OGs).

In addition, ProtozoaDB presented the best species-specific results, with *Entamoeba histolytica* performing 40.77 % of the total OGs (4,086/9,979); *Leishmania infantum* with 27.93 % (2,787/9,979); and *Cryptosporidium hominis* with 19.68 % (1,964/9,979).

### OrthoSearch for Orthologous Database building

Table 1 shows details on how many OGs remained intact and directly migrated to the n-ODs created by our methodology as well as those that were expanded.

After "KO + EggNOG KOG" building, we had a 14.02 % increase in the total number of OGs when compared to KO (16,938/14,856 OGs). In addition, 18.63 % KO

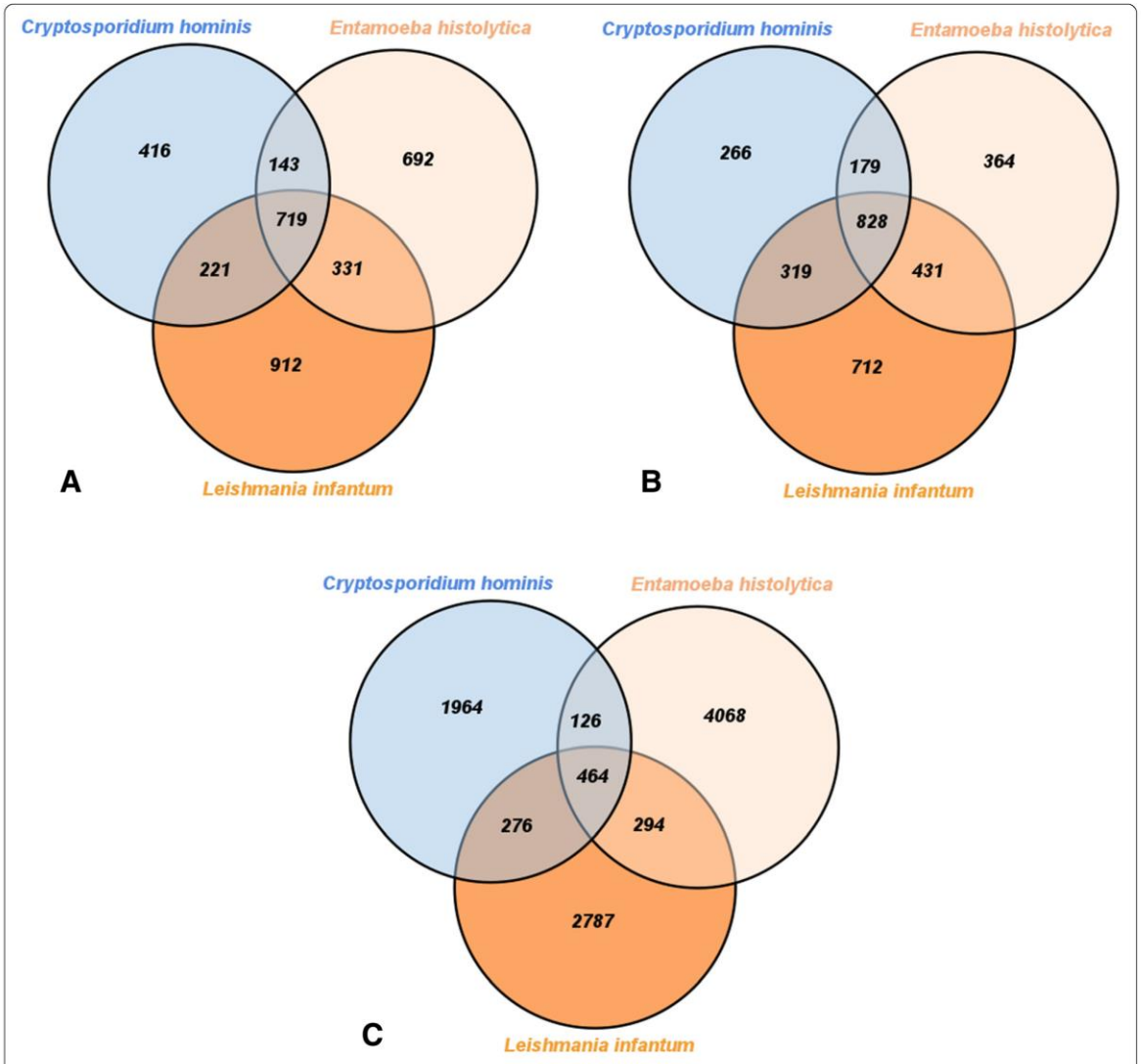


**Fig. 3** Orthologous groups' superposition; A Venn Diagram depicting how many species-specific, pairwise and core orthologous groups were inferred for the protozoan organisms against (**a**) KO, (**b**) EggNOG KOG and (**c**) ProtozoaDB databases

OGs were expanded (2,769/14,856 OGs). When this n-OD was confronted against the impersonated ProtozoaDB OGs, we had a 63.54 % increase in the total number of OGs (27,701/16,938) and 17.88 % "KO + Eggnog KOG" OGs were expanded. Table 2 summarizes how many OGs, proteins, the average proteins per group and each n-OD size. "KO + EggNOG KOG + ProtozoaDB", when compared to KO, provides 86.46 % more OGs (27,701/14,856 OGs) and 22.45 % more proteins (2,582,631/2,109,027 proteins). At last, Table 3

**Table 1** n-OD creation details

| n-OD creation step | | OGs in the OD (i) | OGs in the impersonated proteome (ii) | Intact OGs on the OD(iii) | Intact OGs in the impersonated proteome (iv) | OGs to be expanded(v) | Total OGs in the n-OD (vii) |
|---|---|---|---|---|---|---|---|
| OD name | Impersonated proteome | | | | | | |
| KO | EggNOG KOG | 14856 | 4851 | 12087 | 2082 | 2769 | 16938 |
| KO + EggNOG KOG | ProtozoaDB | 16938 | 13792 | 13909 | 10763 | 3029 | 27701 |

**Table 2** n-ODs main characteristics

| OD/feature | Total OGs | Total proteins | Average proteins per OG | OD size |
|---|---|---|---|---|
| KO + EggNOG KOG | 16938 | 2518449 | 149 | 1.4 GBytes |
| KO + EggNOG KOG + ProtozoaDB | 27701 | 2582631 | 93 | 1.5 GBytes |

shows protozoan species representation at each created n-OD.

### Inferring protozoan orthologs with OrthoSearch and the n-ODs

With these recently created n-ODs, on our second scenario we executed OrthoSearch using as input such n-ODs and the same three protozoan species then compared the obtained results against previous KO analysis. Figure 4 depicts coverage percentage data for each of the OG databases created by the methodology itself, for each organism adopted.

Our methodology provided an 86.47 % increase on the total number of OGs and a 22.45 % on the total amount of proteins when comparing "KO + EggNOG KOG + ProtozoaDB" against KO. Although there was a relevant increase in the number of inferred OGs for *Cryptosporidium hominis* (from 1,499 up to 3,254 groups), coverage increase was very subtle (10 %-12 %). *Entamoeba histolytica*, on the other hand, shows a relevant increase in coverage (19 %), especially when confronted against "KO + EggNOG KOG + ProtozoaDB" n-OD. *Leishmania infantum* had a very similar behavior to *Cryptosporidium hominis*, with a total of up to 4,760 OGs and from 15 % up to 17 % coverage.

We also consolidated the reciprocal best hits obtained in Venn diagrams, so that we might have a glimpse on species-specific, pair-to-pair and core OGs, as shown in Fig. 5.

Additional file 4 shows how many OGs were inferred as best hits with OrthoSearch when protozoan species were confronted against each of the generated n-ODs. "KO + EggNOG KOG" provided the best OGs core coverage ratio – 18.76 % (784/4,178) and "KO + EggNOG KOG + ProtozoaDB", 5.65 % (627/11,089). "KO + EggNOG KOG + ProtozoaDB" had the best results in species-specific OGs, with *Entamoeba histolytica* at a 36.80 % ratio (4,081/11,089); *Leishmania infantum* with 29.94 % (3,320/11,089); and *Cryptosporidium hominis* at 18.81 % (2,086/11,089).

### Comparison between the n-ODs and OrthoMCLDB

After submitting the same protozoan species to OrthoMCLDB online phyletic pattern search, 3,516 (*Cryptosporidium hominis*), 6,107 (*Entamoeba histolytica*) and 7,538 (*Leishmania infantum*) OGs were inferred. OrthoMCLDB inferred a 760 OGs core, which represents 5.12 % of the total best hits (760/14,846).

Concerning species-specific OGs, OrthoMCLDB detected 2,340 – 15.76 % (2,340/14,846) OGs for *Cryptosporidium hominis;* 4,816 – 32.44 % (4,816/14,846) for *Entamoeba histolytica*; and 6,145 – 41.39 % for *Leishmania infantum*; at last, pairwise shared OGs corresponded to 162 (*Cryptosporidium hominis* and *Entamoeba histolytica*), 254 (*Cryptosporidium hominis* and *Leishmania infantum*) and 369 (*Entamoeba histolytica* and *Leishmania infantum*) OGs respectively. Figure 6 shows a Venn diagram with obtained results.

### Potential *Leishmania* spp. targets against the human genome

A BlastP against our largest created n-OD, "KO + Eggnog KOG + ProtozoaDB" (27,701 orthologous groups) allowed us to infer 7,622 (27.5 %) orthologous groups which did not perform any hit against the human proteome. Among such, 6.5 % (1,805/27,701) groups belong to KO or Eggnog KOG, but are not available in ProtozoaDB, which contains only protozoan organisms (*Leishmania* spp. included). Furthermore, 13 orthologous groups (0.05 %) contain at least one *Leishmania* spp. (Table 4), that should be considered as potential targets for further analysis.

The same BlastP query against each of the original ODs provided us the results listed in Table 5. These groups have no similarity with the human proteome and have at least one *Leishmania* spp. sequence.

## Discussion

In this analysis, we adopted new programming languages and updated the OrthoSearch pipeline with several bioinformatics tools, rewriting it to be later used in homology inference analyses and n-ODs creation.

OrthoSearch uses an algorithm based on reciprocal best hits calculation via HMM profiles, with Mafft being

**Table 3** Protozoan species contribution for each n-OD

| OD | Total OGs | OGs with at least one protozoan species | Ratio | Total OD proteins | Protozoan proteins | Ratio |
|---|---|---|---|---|---|---|
| KO | 14856 | 3612 | 24.31 % | 2108653 | 46027 | 2.18 % |
| KO + EggNOG KOG | 16938 | 5851 | 34.54 % | 2518449 | 74630 | 2.96 % |
| KO + EggNOG KOG + ProtozoaDB | 27701 | 17305 | 62.47 % | 2582631 | 138814 | 5.37 % |

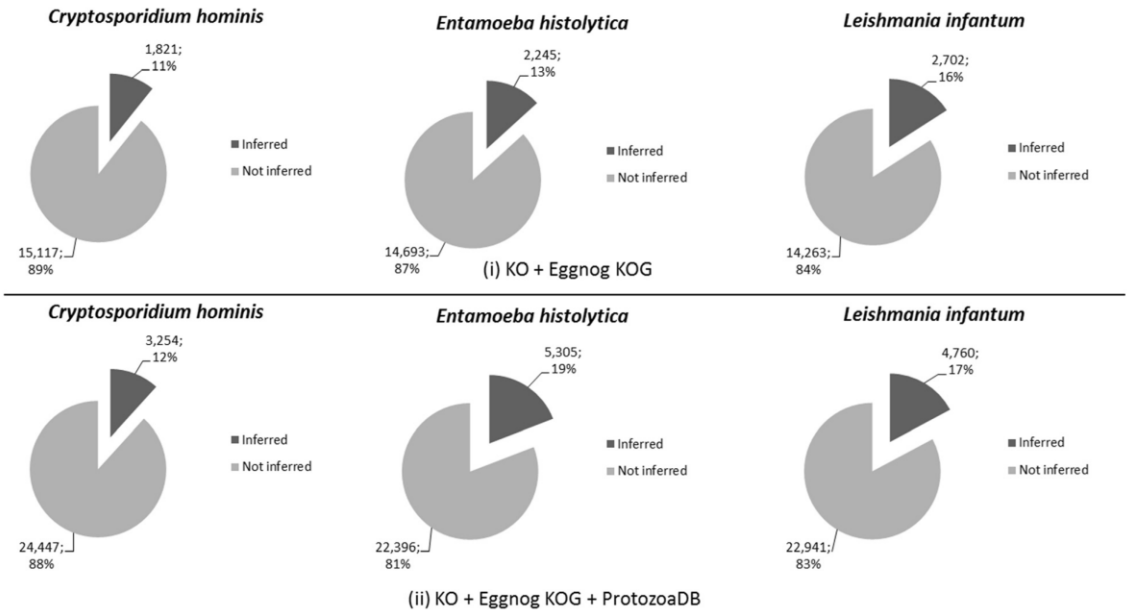


**Fig. 4** OrthoSearch inferred orthologous groups and coverage, per organism, with the databases created by the methodology itself; A detailed view on how many orthologous groups were inferred with (i) "KO + EggNOG KOG" and (ii) "KO + Eggnog KOG + ProtozoaDB" databases and what do such numbers represent against the organisms total protein numbers

responsible for providing MSA's and HMMER3 tools for generating HMM models, profiles and statistics. It benefits from multithreading provided by both Mafft and HMMER3 and accepts any of both tools available parameters. In a regular run, OrthoSearch uses an e-value cut-off flexible enough in order to aid on later profiles calculation.

Each of the studied ODs has its particular characteristics (e.g., while KO contains OGs from all life domains, EggNOG KOG contains only eukaryotes OGs and ProtozoaDB only protozoan OGs). That might influence the obtained results when running OrthoSearch with such ODs and organisms.

OrthoSearch execution with KO OD provides a significantly small core compared to KO size and the total number of best hits. That could be explained as KO contains proteins from many evolutionarily distant organisms, what could pose a challenge in the identification of closely related OGs.

Later, EggNOG KOG OD provided a discrete increase in the obtained protein core, most likely due to EggNOG KOG having only eukaryotic organisms' data. While ProtozoaDB OD provided the smallest core among the three ODs, the total number of species-specific protein is extremely higher. This could be due to the reduced number of species in ProtozoaDB OD, along with the fact that all of those are protozoan organisms. Basically, the odds of obtaining a hit with a protein belonging to the own species being analyzed within OrthoSearch could increase.

We opted to choose a representative protein for each OG at the confronted OD and impersonate an organism multifasta protein data because that could minimize the required computational power and time needed to run OrthoSearch analyses.

Since an OD contains several OGs, which also contains several proteins, that would easily escalate the required time to confront such ODs. In addition, as each OG contains two or more proteins usually from closely related organisms, that could imply the possibility of two (or more) distinct proteins from the same OG obtaining a hit with distinct OGs at the confronted OD.

Our scenarios for n-OD creation were based on KO, EggNOG KOG and ProtozoaDB ODs. According to the literature, each of these ODs were created through particular methodologies: the use of metabolic pathways (KO), heuristic approaches and Gene Ontology [49] support (EggNOG KOG) and OrthoMCL algorithm (ProtozoaDB).

Our methodology allowed us to create n-ODs that either contain intact OGs which originated from the source or the confronted ODs or expanded OGs from the obtained reciprocal best hits inferred by OrthoSearch. The intact OGs contribution relates to offering more OGs for further analyses, while expanded

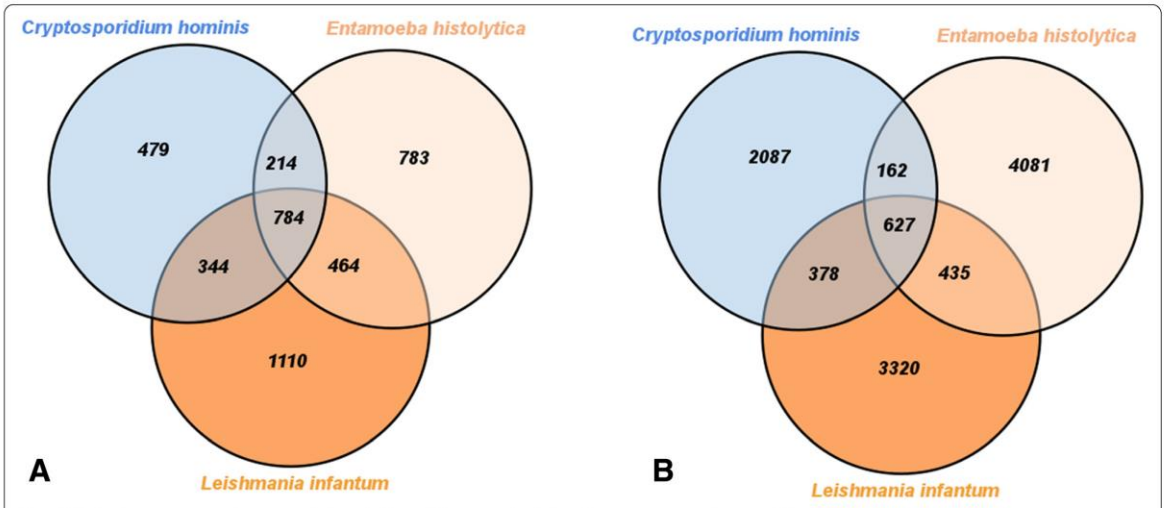


**Fig. 5** Orthologous groups' superposition; A Venn Diagram depicting how many species-specific, pairwise and core orthologous groups were inferred for the protozoan organisms against (**a**) "KO + EggNOG KOG" and (**b**) "KO + EggNOG KOG + ProtozoaDB" n-ODs

ones provide more variability than those OGs from the original databases.

Besides providing a means to improve ProtozoaDB orthology inference, we opted to begin our n-OD creation tasks with KO and Eggnog due to both database variability – proteins from organisms from all life domains – and size. We also decided to maintain the original orthologous groups database identifiers, as well as their functional annotation. That might ease further steps related to information provenance.

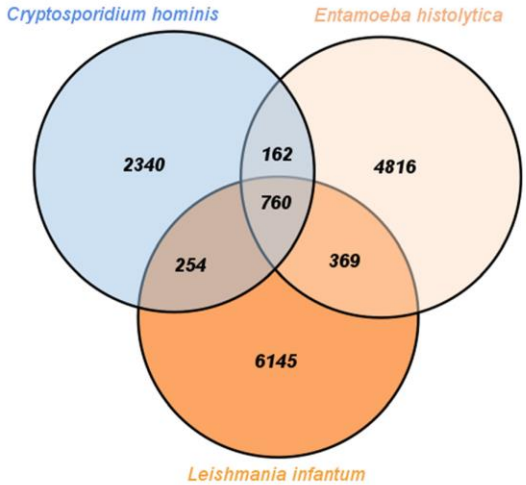


**Fig. 6** Orthologous groups' superposition; A Venn Diagram depicting how many species-specific, pairwise and core orthologous groups were inferred for the protozoan organisms against OrthoMCLDB

Our proposed methodology works as a non-intrusive approach to a HMM-based pipeline – OrthoSearch – without changing its core functions. It uses ODs as input data and is capable to create n-ODs without requiring extensive computational power.

When looking at how protozoan species data fit to our proposed n-OD creation methodology, we observe a 61.98 % increase from KO to "KO + EggNOG KOG" OD, scaling from 3,612 up to 5,851 OGs with at least one Protozoa protein. In addition, the total number of protozoan proteins had a 162 % increase (from 46,027 up to 74,630) in such n-OD.

**Table 4** Potential *Leishmania* spp. targets

| Orthologous group | Annotation |
|---|---|
| K14118.cdhit | Energy-converting hydrogenase B subunit I |
| K13666.cdhit | UDP-GlcNAc:polypeptide alpha-N-acetylglucosaminyltransferase |
| K03668.cdhit | Heat shock protein HslJ |
| K08907.cdhit | Light-harvesting complex I chlorophyll a/b binding protein 1 |
| K13690.cdhit | Alpha-1,3-mannosyltransferase |
| K13674.cdhit | Phosphoglycan alpha-1,2-arabinopyranosyltransferase |
| K08276.cdhit | Ecotin |
| K02817.cdhit | PTS system, trehalose-specific IIB component |
| K01833.cdhit | Trypanothione synthetase/amidase |
| K03329.cdhit | Hypothetical protein |
| K01973.cdhit | Mitochondrial RNA editing ligase 1 |
| K03356.cdhit | Anaphase-promoting complex subunit 9 |
| K13672.cdhit | Galactofuranosyltransferase |

**Table 5** *Leishmania* spp. orthologous groups with no hit against the human proteome (Original orthologous databases)

| Orthologous database | Orthologous groups |
|---|---|
| KO | 5 |
| Eggnog KOG | 1 |
| ProtozoaDB | 1524 |

Furthermore, when KO and "KO + EggNOG KOG + ProtozoaDB" ODs were compared, we identified a 379 % increase in the number of OGs that contain at least one Protozoa protein (from 3,612 up to 17,305) and a 300 % increase in the total number of protozoan proteins (from 46,027 up to 138,814).

A broader dataset is usually desirable, as it may increase the odds of obtaining hits while inferring homology. As more organisms contribute to a n-OD, one might be able to obtain more hits with regular OrthoSearch runs confronting n-ODs and organisms multifasta protein data.

OrthoSearch uses a Markov chain based approach in order to create the n-OD OGs, which tend to comprise more evolutionary distant orthologous proteins than BLAST-based methodologies, such as OrthoMCLDB.

With the n-ODs created within our methodology, our initiative is another step to reinforce possibilities to build a gold, reference dataset [50] for orthology inference.

As more OGs (and respective proteins) from distinct species are added up to the n-ODs created, the methodology offers a broader dataset, with more data variability. Such data may be used for further homology inference analyses, which is a very desirable aspect in several comparative genomics applications. For example, phylogenomic studies which try to address gene conflicts and allow for optimal tree construction [51] or even review species' definition [52].

The obtained results point towards the success of our proposed methodology, which encourage us in refining and creating more n-ODs. Such n-ODs might also be used in order to improve future functional annotation, re-annotation and also potential targets identification.

OrthoMCLDB was able to provide a broader species-specific OGs dataset, varying from 12 % *(Cryptosporidium hominis* - 2,340/2,086) up to 85 % *(Leishmania infantum* – 6,145/3,320) more OGs than our methodology. That may be due to several OrthoMCLDB groups containing only in-paralogs (20,853/124,740) from a single species lineage [35].

When looking at pairwise groups, there is a change in this scenario. Our methodology either provides the same quantitative results as OrthoMCLDB (*Cryptosporidium hominis* and *Entamoeba histolytica* – 162 OGs) or better, with 17.88 % more OGs (*Entamoeba histolytica* and *Leishmania infantum* – 435/369) and up to 48.81 % more OGs (*Cryptosporidium hominis* and *Leishmania infantum* – 378/254).

OrthoMCLDB inferred a larger absolute number of OGs in the three protozoan species core (760) than our methodology (627), which may be related to a broader seed of OGs (124,740/27,701). On the other hand, OrthoMCLDB performed poorly in coverage aspect, with only 0.06 % OGs in its core (760/124,740), while our methodology covered 2.26 % OGs (627/27,701), with our largest n-OD ("KO + EggNOG KOG + ProtozoaDB"). This may be due to the fact that the studied species are not so closely related. In addition, OrthoMCLDB uses a Blast-based algorithm, in a less sensitive approach than our methodology (OrthoSearch uses a protein-profile comparison) [9, 32].

Our methodology provides means for improved orthologous database creation using a HMM-based approach. Those new databases may contain a greater set of evolutionary distant homologous proteins, which could further extend the odds of inferring knowledge regarding the target organisms.

Specifically, our analyses allowed for a better comprehension on three protozoan species, as well as a deeper analysis on potential targets. For example, the obtained protozoan core orthologous proteins may allow us to evaluate which of these are housekeeping proteins and how they relate to the organism fitness.

Also, the species specific proteins – those which do not belong to the core, or those shared between two of the three studied protozoan organisms might be explored either as species-specific or group-specific targets, respectively.

The obtained BlastP results allowed us to infer orthologous groups which contain protozoan proteins – specifically *Leishmania* spp. - that could be used as potential targets for further analysis, as they posed no hit against the human proteome.

Among the *Leishmania* spp. inferred orthologous groups without hits against the human proteome (Table 4) are proteins already described in the literature as possible drug targets, briefly: trypanothione [53] (K01833.cdhit) – which relates to defense against oxidative stress [54]; and alpha-1,3-mannosyltransferase [55, 56] (K13690.cdhit) – enzyme essential to add mannose on the glycosylphosphatidyl, relates to the growing resistance to miltefosine. However, there are also other proteins, not yet described as drug targets, which should be further studied, briefly: the energy-converting hydrogenase B subunit I [55] (K14118.cdhit), found in the Archaea organism *Methanothermobacter thermautotrophicus*, which belongs to a domain related to MnhB subunit of Na+/H+ antiporter and is predicted as an integral membrane protein [57, 58]; and galactofuranosyltransferase [59] (K13672.cdhit), related to the LPG1 gene, which acts as a major ligand for macrophage adhesion [60, 61].

Our methodology also provided means to allocate new and evolutionary distant proteins to the original orthologous groups' databases, identifying orthology relationships which have not been previously described.

Even though this is a preliminary analysis, it allowed us to evaluate the applied methodology and to forecast how its results may be used for protozoan target identification, either in a species-specific or shared point-of-view. This methodology will be later applied to all of 22 ProtozoaDB [46] protozoan organisms.

## Conclusions

Our analyses used initially KO and EggNOG KOG databases as a starting point, adding later ProtozoaDB. We were able to create two n-ODS, "KO + EggNOG KOG" and "KO + EggNOG KOG + ProtozoaDB", with each providing a larger amount of OGs when compared to the original. Those represent a broader dataset that can be used in future homology inference, annotation transfer and protozoan targets identification.

So far, our methodology allowed the identification of 13 potential targets in protozoan that would not have been identified without the distant homology approach provided by OrthoSearch and the n-ODs created by our methodology.

## Additional files

**Additional file 1: Orthologous databases main characteristics.** This file provides a detailed view on the orthologous databases characteristics, regarding: how many organisms (total and protozoan only), orthologous groups, total proteins and the database total size. (XLS 6 kb)

**Additional file 2: Sample impersonated multifasta protein sample; This file provides a quick view on how the organisms' representative proteins were stored in a single multifasta file, which impersonates an organism proteome and actually contains proteins from several orthologous groups from a single orthologous database.** (FASTA 1 kb)

**Additional file 3: "KO + Eggnog KOG + ProtozoaDB" orthologous groups (protein identifiers only).** For each ortholog group, we list all protein identifiers (corresponding organism and accession number). (BZ2 9971 kb)

**Additional file 4: Inferred RBH between protozoan organisms and orthologous databases.** This file provides information on inferred RBH between protozoan species against KO database and the two n-ODs created by the methodology as well. It is detailed on species-specific, pairwise and core RBH (shared between all organisms) absolute RBH and their percentage when compared to the total amount of obtained RBH. (XLS 27 kb)

#### Abbreviations

KO: Kegg Orthology; OG: orthologous group; OD: orthologous database; n-OD: new orthologous database.

#### Competing interests

The authors declare that there is no conflict of interest regarding the publication of this paper.

#### Authors' contributions

AMRD designed and coordinated the analyses. NPKF and RJ were responsible for programming tasks, experiment design and writing the manuscript. NPKF performed and collected experiment data. All authors revised the final version of the manuscript and AMRD approved it.

#### Acknowledgements

Our sincere thanks to everyone involved, not only from our laboratory, but to everyone who inspired, stimulated and provided us with feedback. To Salvador Capella-Gutierrez (CRG) for the script to obtain representative sequences from multiple alignments. A special thank you to our fellow colleagues: Diogo Tschoeke, Rafael Cuadrat and Sérgio Serra.

Received: 2 June 2015 Accepted: 11 September 2015
Published online: 29 September 2015

#### References

1. Imam T. The complexities in the classification of protozoa: a challenge to parasitologists. Bayero J Pure Appl Sci. 2009;2:159–64.
2. Cavalier-Smith T. Predation and eukaryote cell origins: A coevolutionary perspective. Int J Biochem Cell Biol. 2009;41:307–22 [Molecular and Cellular Evolution: A Celebration of the 200th Anniversary of the Birth of Charles Darwin].
3. Cavalier-Smith T. Kingdoms Protozoa and Chromista and the eozoan root of the eukaryotic tree. Biol Lett. 2009;6(3):rsbl20090948.
4. Cavalier-Smith T. Kingdom Protozoa and Its 18 Phyla. Microbiol Rev. 1993;57:953–94.
5. Widmer G, London E, Zhang L, Ge G, Tzipori S, Carlton J, et al. Preliminary Analysis of the Cryptosporidium muris Genome. In: Ortega-Pierres G, Cacciò Simone M, Fayer R, Mank TG, Smith HV, Thompson RCA, editors. GIARDIA AND CRYPTOSPORIDIUM From Molecules to Disease. Wallingford, UK: CAB International; 2009. p. 320–7.
6. Thompson RCA. The Impact of Giardia on Science and Society. In: Ortega-Pierres, Guadalupe; Cacciò, Simone M; Fayer, Ronald; Mank, Theo G; Smith, Huw V; Thompson R, editors. GIARDIA AND CRYPTOSPORIDIUM From Molecules to Disease. Wallingford, UK: CAB International; 2009. p. 1–11.
7. Pain A, Renauld H, Berriman M, Murphy L, Yeats C, Weir W, et al. Genome of the host-cell transforming parasite Theileria annulata compared with T. parva. Science. 2005;309:131–3.
8. Elmore S, Jones JL, Conrad P, Patton S, Lindsay DS, Dubey JP. Toxoplasma gondii: epidemiology, feline clinical aspects, and prevention. Trends Parasitol. 2010;26:190–6.
9. Carlton JM, Adams JH, Silva JC, Bidwell SL, Lorenzi H, Caler E, et al. Comparative genomics of the neglected human malaria parasite Plasmodium vivax. Nature. 2008;455:757–63.
10. Brayton K, Lau AOT, Herndon DR, Hannick L, Kappmeyer LS, Berens SJ, et al. Genome sequence of Babesia bovis and comparative analysis of apicomplexan hemoprotozoa. PLoS Pathog. 2007;3:1401–13.
11. WHO | Neglected Diseases [http://www.who.int/neglected_diseases/diseases/en/].
12. WHO/Department of control of neglected tropical diseases. Investing to Overcome the Global Impact of Neglected Tropical Diseases. Geneva, Switzerland: World Health Organization; 2015.
13. The London declaration on Neglected Tropical Diseases [http://unitingtocombatntds.org/sites/default/files/resource_file/london_declaration_on_ntds.pdf].
14. WHO | Leishmaniasis [http://www.who.int/mediacentre/factsheets/fs375/en/].
15. WHO Technical Report Series [http://whqlibdoc.who.int/trs/WHO_TRS_949_eng.pdf].
16. Singh N, Chikara S, Sundar S. SOLiD™ sequencing of genomes of clinical isolates of leishmania donovani from india confirm leptomonas co-infection and raise some key questions. PLoS One. 2013;8, e55738.
17. Brotherton M-C, Bourassa S, Leprohon P, Légaré D, Poirier GG, Droit A, et al. Proteomic and genomic analyses of antimony resistant leishmania infantum mutant. PLoS One. 2013;8, e81899.
18. Tschoeke DA, Nunes GL, Jardim R, Lima J, Dumaresq AS, Gomes MR, et al. The comparative genomics and phylogenomics of leishmania amazonensis parasite. Evol Bioinforma Online. 2014;10:131–53.
19. Hardison RC. Comparative genomics. PLoS Biol. 2003;1:156–60.
20. Koonin EV. Orthologs, paralogs, and evolutionary genomics. Annu Rev Genet. 2005;39:309–38.
21. Sequence - Evolution - Function - NCBI Bookshelf [http://www.ncbi.nlm.nih.gov/books/NBK20260/].

22. Dessimoz C, Gabaldón T, Roos DS, Sonnhammer ELL, Herrero J, Quest for Orthologs Consortium. Toward community standards in the quest for orthologs. Bioinforma Oxf Engl. 2012;28:900–4.
23. Delsuc F, Brinkmann H, Philippe H. Phylogenomics and the reconstruction of the tree of life. Nat Rev Genet. 2005;6:361–75.
24. Li L, Stoeckert CJ, Roos DS. OrthoMCL: Identification of ortholog groups for eukaryotic genomes. Genome Res. 2003;13:2178–89.
25. Dessimoz C. Editorial: Orthology and applications. Brief Bioinform. 2011;12:375–6.
26. Kristensen DM, Wolf YI, Mushegian AR, Koonin EV. Computational methods for Gene Orthology inference. Brief Bioinform. 2011;12:379–91.
27. Cuadrat RRC, Cruz SM da S, Tschoeke DA, Silva E, Tosta F, Jucá H, et al. An Orthology-Based Analysis of Pathogenic Protozoa Impacting Global Health: An Improved Comparative Genomics Approach with Prokaryotes and Model Eukaryote Orthologs. OMICS J Integr Biol. 2014;140624130015005.
28. Dewey CN. Positional orthology: putting genomic evolutionary relationships into context. Brief Bioinform. 2011;12:401–12.
29. Dalquen DA, Altenhoff AM, Gonnet GH, Dessimoz C. The Impact of Gene Duplication, Insertion, Deletion, Lateral Gene Transfer and Sequencing Error on Orthology Inference: A Simulation Study. PLoS One. 2013;8, e56925.
30. Da Cruz SMS, Batista V, Silva E, Tosta F, Vilela C, Cuadrat R, et al. Detecting distant homologies on protozoans metabolic pathways using scientific workflows. Int J Data Min Bioinforma. 2010;4:256–80.
31. Remm M, Storm CEV, Sonnhammer ELL. Automatic clustering of orthologs and in-paralogs from pairwise species comparisons. J Mol Biol. 2001;314:1041–52.
32. DeLuca TF, Cui J, Jung J-Y, Gabriel KCS, Wall DP. Roundup 2.0: Enabling comparative genomics for over 1800 genomes. Bioinformatics. 2012;28:715–6.
33. Wall DP, Fraser HB, Hirsh AE. Detecting putative orthologs. Bioinformatics. 2003;19:1710–1.
34. Rasmussen MD, Kellis M. A Bayesian Approach for Fast and Accurate Gene Tree Reconstruction. Mol Biol Evol. 2011;28:273–90.
35. Chen F, Mackey AJ, Stoeckert CJ, Roos DS. OrthoMCL-DB: querying a comprehensive multi-species collection of ortholog groups. Nucleic Acids Res. 2006;34 suppl 1:D363–8.
36. O'Brien KP, Remm M, Sonnhammer ELL. Inparanoid: a comprehensive database of eukaryotic orthologs. Nucleic Acids Res. 2005;33(Database issue):D476–80.
37. Tatusov RL, Fedorova ND, Jackson JD, Jacobs AR, Kiryutin B, Koonin EV, et al. The COG database: an updated version includes eukaryotes. BMC Bioinformatics. 2003;4:41.
38. Powell S, Forslund K, Szklarczyk D, Trachana K, Roth A, Huerta-Cepas J, et al. eggNOG v4.0: nested orthology inference across 3686 organisms. Nucleic Acids Res 2013;42(D1):gkt1253.
39. Powell S, Szklarczyk D, Trachana K, Roth A, Kuhn M, Muller J, et al. eggNOG v3.0: orthologous groups covering 1133 organisms at 41 different taxonomic ranges. Nucleic Acids Res. 2011;40:D284–9.
40. Da Cruz SMS, Batista V, Dávila AMR, Silva E, Tosta F, Vilela C, et al. OrthoSearch: a scientific workflow approach to detect distant homologies on protozoans. New York, NY, USA: ACM; 2008. p. 1282–6 [SAC '08].
41. Taylor IJ, Deelman E, Gannon DB, Shields M. Workflows for E-Science. 2007.
42. Timmers LFSM, Pauli I, Barcellos GB, Rocha KB, Caceres RA, de Azevedo WF, et al. Genomic databases and the search of protein targets for protozoan parasites. Curr Drug Targets. 2009;10:240–5.
43. S SK, R.K. G, Ghosh M. Comparative in-silico genome analysis of Leishmania (Leishmania) donovani: A step towards its species specificity. Meta Gene. 2014;2:782–98.
44. KEGG Orthology [http://www.genome.jp/kegg/ko.html].
45. KEGG. Kyoto Encyclopedia of Genes and Genomes [http://nar.oxfordjournals.org/content/28/1/27.abstract?ijkey=8863b3d5385d8722cdd87f19eccd8a62b567a0b4&keytype2=tf_ipsecsha].
46. Dávila AMR, Mendes PN, Wagner G, Tschoeke DA, Cuadrat RRC, Liberman F, et al. ProtozoaDB: dynamic visualization and exploration of protozoan genomes. Nucleic Acids Res. 2008;36(Database issue):D547–52.
47. Camacho C, Coulouris G, Avagyan V, Ma N, Papadopoulos J, Bealer K, et al. BLAST+: architecture and applications. BMC Bioinformatics. 2009;10:421.
48. RefSeq: NCBI Reference Sequence Database [http://www.ncbi.nlm.nih.gov/refseq/].
49. Ashburner M, Ball CA, Blake JA, Botstein D, Butler H, Cherry JM, et al. Gene Ontology: tool for the unification of biology. Nat Genet. 2000;25:25–9.
50. Chen F, Mackey AJ, Vermunt JK, Roos DS. Assessing Performance of Orthology Detection Strategies Applied to Eukaryotic Genomes. PLoS One. 2007;2.
51. McCormack JE, Faircloth BC, Crawford NG, Gowaty PA, Brumfield RT, Glenn TC. Ultraconserved elements are novel phylogenomic markers that resolve placental mammal phylogeny when combined with species-tree analysis. Genome Res. 2012;22:746–54.
52. Konstantinidis KT, Tiedje JM. Genomic insights that advance the species definition for prokaryotes. Proc Natl Acad Sci. 2005;102:2567–72.
53. Colotti G, Baiocco P, Fiorillo A, Boffi A, Poser E, Chiaro FD, et al. Structural insights into the enzymes of the trypanothione pathway: targets for antileishmaniasis drugs. Future Med Chem. 2013;5:1861–75.
54. Krauth-Siegel RL, Meiering SK, Schmidt H. The Parasite-Specific Trypanothione Metabolism of Trypanosoma and Leishmania. Biol Chem. 2003;384.
55. Shinde S, Mol M, Jamdar V, Singh S. Molecular modeling and molecular dynamics simulations of GPI 14 in Leishmania major: insight into the catalytic site for active site directed drug design. J Theor Biol. 2014;351:37–46.
56. Garami A, Ilg T. The role of phosphomannose isomerase in Leishmania mexicana glycoconjugate synthesis and virulence. J Biol Chem. 2001;276:6566–75.
57. Ito M, Guffanti AA, Krulwich TA. Mrp-dependent Na+/H+ antiporters of Bacillus exhibit characteristics that are unanticipated for completely secondary active transporters. FEBS Lett. 2001;496:117–20.
58. Hiramatsu T, Kodama K, Kuroda T, Mizushima T, Tsuchiya T. A putative multisubunit Na+/H+ antiporter from Staphylococcus aureus. J Bacteriol. 1998;180:6642–8.
59. Huang C, Turco SJ. Defective galactofuranose addition in lipophosphoglycan biosynthesis in a mutant of Leishmania donovani. J Biol Chem. 1993;268:24060–6.
60. Zhang K, Barron T, Turco SJ, Beverley SM. The LPG1 gene family of Leishmania major. Mol Biochem Parasitol. 2004;136:11–23.
61. Spath GF, Epstein L, Leader B, Singer SM, Avila HA, Turco SJ, et al. Lipophosphoglycan is a virulence factor distinct from related glycoconjugates in the protozoan parasite Leishmania major. Proc Natl Acad Sci. 2000;97:9258–63.

## 8.4 Anexo D – Artigo 4 – "A Multi-Platform Workflow Management System for supporting e-Science experiments"

Carrión A, Caballer M, Blanquer Ignacio, **Kotowski N**. **"A Multi-Platform Workflow Management System for supporting e-Science experiments".**

**Este artigo foi submetido para a revista *Journal of Systems and Software*.**

***A Multi-Platform Workflow Management System for supporting e-Science experiments***

Revista: *Journal of Systems and Software*
Tipo de manuscrito: *Research paper*
Data de submissão: 27 de Abril de 2015
Autores: Carrión, Abel; Caballer, Miguel; Blanquer, Ignacio; Kotowski, Nelson

Identificador do manuscrito: JSS-D-15-00341

Associado ao objetivo específico 2.2.3, este artigo detalha o projeto e construção da máquina de *workflow* criada pelo discente Abel Carrión, do I3M/UPV, durante o período de doutorado sanduíche no exterior do autor da tese.

# A Multi-Platform Workflow Management System for supporting e-Science experiments

A.A. Carrión[a,1], M. Caballer[1,1], I. Blanquer[a,1,1], N. Kotowski[b,1], A.M.R. Dávila[b,1]

[a] *Instituto de Instrumentación para Imagen Molecular (IM3). Centro mixto CSIC - Universitat Politècnica de València - CIEMAT. Camino de Vera s/n, 46022 Valencia, España.*

[b] *Computational and Systems Biology Laboratory. Oswaldo Cruz Institute, Rio de Janeiro. RJ, Brazil, 21040-360*

**Abstract**

Scientific research is going through a data deluge crisis in which the amount of data generated is reaching the order of terabytes per day. Up to this point, e-Science and the modelling of its processes with Scientific Workflows has enabled the discovery of new research facts. However, in a data flow scenario the execution of Scientific Workflows is a task with many details that must handled by software programs called Workflow Management Sys- tems (WMSs) by using all the available computing and storage resources. Moreover, the recent emergence of a new distributed computing paradigm, Cloud Computing, offers new possibilities for addressing the processing of these huge datasets. As a consequence, legacy WMSs have been updated to support cloud computing but due to their original approach (mostly Grid Computing) are not optimized to use key features of it. For that reason, in this work we detail the design and implementation of a multi-platform WMS optimized to use clusters and cloud computing infrastructures. The proposed tool has been tested using a comparative genomics workflow called Orthosearch, obtaining promising results.

*Keywords:* Workflow, Workflow Management Systems, Multi-platform, e-Science, Cloud Computing, Comparative genomics

*Email addresses:* abcarcol@i3m.upv.es (A.A. Carrión), micafer@i3m.upv.es (M. Caballer), iblanque@dsic.upv.es (I. Blanquer), nelsonpeixoto@fiocruz.br (N. Kotowski), davila@fiocruz.br (A.M.R. Dávila)

*Preprint submitted to Journal of Systems and Software27th April 2015*

## 1.Introduction

Traditional science is representative of two different philosophical trends within the history of science, theoretical/analytical and experimental/observational. But, in the last decades, Computer Science has revolutionized the way in which science and engineering are conducted and nowadays is recognized as the “third branch” of science along with theory and experiment [1]. With the inclusion of computation, the term e-Science was coined as “the application of computer technology to the undertaking of modern scientific investigation, including the preparation, experimentation, data collection, results dissem- ination, and long-term storage and accessibility of all materials generated through the scientific process” [2]. In short, e-Science is the Science in which the use of computers becomes indispensable for performing scientific research in an efficient way.

The relation between Science and computing goes back to the 1960s, when powerful computers (in terms of speed calculation), called supercomputers, were introduced for performing scientific and engineering problems. At that time, a typical experimental scenario consisted in a repetitive cycle of moving data to a supercomputer for processing, submitting the executions and retrieving the outputs from the data storage [3]. Obviously, this process had to be automated for allowing scientists to focus on their research and not in the computational management. Fortunately, at the same time, the business community was addressing how to automate business processes and as a result the *Workflow* concept was born. In the business context, a *Workflow* can be defined as the orchestration of a set of activities in order to accomplish a larger and sophisticated goal. A specialization of this idea was adopted by the research community to model e-Science processes, the Scientific Workflows (SWFs). In this programming model, scientific applications are described as a set of tasks that have dependencies between them. It means that a task will start its execution only when the tasks it depends on have completed their execution.

The execution of workflow applications is a task with many details. A typical workflow is composed of hundreds of tasks that must be executed in a coordinated way. Moreover, all these tasks must be submitted to specific computing resources and the required inputs must be made available to the application. In data intensive applications, the staging of the input files could require transferring huge amounts of data between resources. In a complex scenario like this one, it is possible to identify

several single points of failure: the reception of user inputs, the data transfer between tasks, task executions, hardware crashes, etc. Thus, in these cases it is necessary to carry out actions for resuming the execution, such as retrying the data transfer, rescheduling the task or resetting the resources. Software in charge of dealing with all these aspects are called Workflow Management Systems.

As new computing paradigms emerge and infrastructure evolve, so do the WMSs that support these computing back-ends. Traditionally, Scientific Workflow Applications have been extensively deployed in high-performance computing infrastructures, such as powerful clusters and supercomputers. Later, a highly distributed infrastructure, the Grid, appeared as an alternative to traditional approaches. Grid Computing offered secure and collaborative resource sharing across multiple, geographically distributed institutions. Due to the high impact of Grid infrastructures on the research community, the definition of e-Science was revised as “computationally intensive science that is carried out in highly distributed network environments, or science that uses immense data sets”. In the last years, a new distributed computing paradigm, Cloud Computing, has appeared as another viable [4] platform for running scientific applications. In fact, some of their main features, such as rapid elasticity, resource pooling, and pay per use, are well suited to the nature of scientific applications that experience a variable demand during its execution.

Because the scientific experimentation is suffering from a data deluge phe nomenon where the amount of data generated is reaching the order of terabytes per day, a huge amount of resources are needed to process this data and enable research. For that reason, it is crucial that WMSs have multi-platform support. Although current WMSs already support various platforms for the execution of workflow applications, many desirable features of cloud computing are not implemented, such as the dynamic provisioning of resources. This is because current WMSs are derived from grid computing projects and thus are optimized for grids [5]. For that reason, in this work we present a WMS with multi-platform support and optimized to execute workflow applications in cloud computing platforms and clusters.

The remainder of the paper is structured as follows. Firstly, Section 2 offers a survey of the main infrastructures used for executing a scientific workflow application. Then, Section 3 gives an overview of the state-of-the-art WMSs. Afterwards, Section 4 explains in detail the architecture of a novel multi-platform WMS with actual support

of cloud computing resources. Using as use case a bioinformatics pipeline, called Orthosearch, Section 6 explains the experiments carried out with our WMS and the results obtained. Finally, conclusions and future working lines are exposed.

## 2. Computing Platforms survey

The aim of this section is to justify the need of multi-platform support in current WMSs by showing the differences between distributed computing paradigms (supercomputing, cluster computing, grid computing and cloud computing) when executing scientific workflow applications.

Since the introduction of the first supercomputers, workflow applications have been extensively deployed in these high-performance computing (HPC) infrastructures. At that time, HPC was only available to institutions that could afford the expensive cost of these machines. This fact leads to the apparition of cluster computing, offering a cheap way for gaining access to significant computing power. On these infrastructures, the priority of a workflow execution was to minimize the response time by maximizing the utilization of the resources available for the workflow. With the popularity of Internet, the availability of powerful computers and high-speed network technologies, grids became available and were also used for workflow execution. Grid Computing offered secure and collaborative resource sharing across multiple, geographically distributed institutions. Due to the highly-distributed nature of Grid resources, the scheduling process became more complex and data movement across wide distances might be necessary. In order to improve the scheduling process, researchers have formulated many efficient scheduling algorithms (mostly based on heuristics). But, even in this case, the focus was on minimizing the execution time of the workflow. Although grids offered a huge amount of resources, their heterogeneity resulted on users being limited to those resources with a software environment capable of supporting their legacy applications. Obviously, Grid providers cannot support the diversity of all possible environments. However, around the end of 2007, the Cloud Computing paradigm appeared as another viable platform for running scientific applications [6]. In particular, the use of virtualization provides many useful benefits for scientific applications including: customization of the software stack by the user, performance isolation, check-pointing and migration, better reproducibility of scientific analyses, and enhanced support of legacy applications [7] [8]. Other characteristics of the Cloud such as the

elasticity and pay-per-use are well suited to the nature of scientific workflows that experience a variable demand of software and hardware resources during the execution of the different tasks. Because clouds give the perception or illusion of infinite computing resources, the only limitations to the reduction of the execution time are the available resources that the user can afford and the structure of the workflow itself. Therefore, the goal in clouds is to achieve a trade-off between minimizing the execution time and the financial cost.

The purpose of Table 1 is to show that the ideal platform for executing a scientific workflow will depend on the software and hardware requirements of each task and the user profile (some have access to supercomputers, others to grids, etc.). Table 1 shows a comparative between the four platforms mentioned before from the user's point of view.

| | **Supercomputer** | **Clusters** | **Grids** | **Clouds** |
|---|---|---|---|---|
| **Qualiy of Service** | Limited | Limited | Yes | Yes |
| **Resource Allocation** | Centralized | Centralized | Decentralized | Centralized/Decentralize |
| **Resource Handling** | Centralized | Centralized | Distributed | Centralized/Distributed |
| **MPI support** | Full | Full | Partial | Partial |
| **Reliability** | No | No | Medium | Full |
| **Security** | Yes | Yes | Medium | No |
| **User-friendly** | No | No | Medium | Yes |
| **Virtualization** | No | No | No | Yes |
| **Task size** | Single large | Single large | Single large | Small & medium |
| **Heterogeneity** | No | Yes | Yes | Yes |
| **Scalability** | No | No | Medium | Yes |
| **Cost** | High | High | Low | Low |
| **Application** | HPC, HTC | HPC, HTC | HPC, HTC, Batch | HPC,HTC,Batch |

Table 1: Computing platform comparative

Because each platform offers different advantages and disadvantages, there is not an ideal choice in every scenario and, thus, it is crucial that modern WMSs support the execution of a workflow in a mix of computing platforms.

## 3. Related work

Due to the crucial role that workflow applications play in the scientific community, most current WMSs were developed to enable the execution of these applications in grid computing platforms. When Clouds became mainstream, WMSs were enhanced to support it. In this section, we present a brief description of the most prominent WMS found in the state-of-the-art which are related with our work. An extensive overview of many of the most influential workflow management systems developed for executing scientific workflows on distributed infrastructures can be found on [9] [10] [3].

Pegasus [11] is a mature Workflow Management System that combines features such as portability across a wide range of infrastructures, scalability, data management capabilities, exhaustive monitoring and complex workflow restructuring or transformations. It can be used with popular programming languages among the scientific community (such as Java, Python Perl) through its APIs (application programming interfaces) and also supports submission via web portals. Although it supports multiple cloud providers, it does not dynamic provision resources with different hardware and software requirements.

Taverna [12] is a WMS with a strong focus on bioinformatics where all computational workflow steps are Web Services. Workflows can be designed and executed on local desktop machines through the workbench or through other clients or web interfaces using the server mode. The server supports requests from many users to execute remote workflows with support of both grid and cloud platforms. It uses myExperiment [13] as repository for sharing and reusing workflows and the BioCatalogue and Biodiversity catalogue for Web Services discovery.

Kepler [14] is build upon the mature, dataflow-oriented Ptolemy system [15]. From it, Kepler inherits several features such as the GUI and the Actor-Director model. In Kepler, workflows can be created connecting components called Actors which will process the data. An actor has several Ports for expressing input and output data and the director determines the model of computation used by the workflow. To the best of our knowledge, it only supports distributed execution via Web and Grid services, but not Cloud Computing platforms.

Galaxy [16] is an open, web-based approach that facilitates genomics research. It provides a collaborative environment for performing complex analyses, with automatic provenance tracking, allowing the transparent sharing of computational details, intent and context. Its objective is to offer accessible, reproducible and transparent computational research. A Galaxy instance supports running on compute clusters through Portable Batch System (PBS) and Sun Grid Engine (SGE).

Makeflow [17] is a scalable, fault-tolerant WMS for running data-intensive workflows across multiple execution engines (local execution, SGE, HTCondor, Hadoop and a combination of them). One of the main features of Makeflow is that workflow descriptions are generated using a nearly identical syntax of the Unix Make utility for compiling and linking programs. It also produces a transaction log to recover from failures.

The Cloudbus Workflow Engine [18] provides an intuitive workflow for application composition, an XML-based workflow language for structured representation, and an easy-to-use portal with discovery, monitoring and scheduling components. Moreover, it incorporates a market-oriented utility computing model that supports desktop, grid and cloud computing.

Calheiros et al. [5] propose a WMS that supports efficient and automated execution of workflows in clouds with adaptive execution, fault tolerance, and data privacy. Although, the authors detail the requirements of a WMS that supports all these features, they give a vague description of the WMS architecture and its implementation.

The WMS described in this work distinguishes from the previous ones with features such as multi-cloud (public, private, hybrid) and multi-platform support (clusters and clouds). In addition, it presents a novel approach for the on-demand provisioning of cloud resources with ad-hoc hardware and software requirements. The solution will be released under GPL v3 license and it will be available for downloading at github (https://github.com/abel- carrion).

## 4. System architecture

The aim of this section is to describe the design and implementation of the architecture behind the WMS developed. The overall organization of the system is depicted in Figure 1. This schema is based on the one showed in [10], one of the most cited papers about the taxonomy of Grid WMSs. Our architecture is almost identical but extended to support a multi-platform scenario. In fact, our WMS currently supports execution on clusters, public clouds (Amazon EC2, Google Cloud Platform and Microsoft Azure) and private clouds (OpenNebula and OpenStack).

As Figure 1 shows, the structure of the architecture is split into two parts: built-time components and run-time components. On the one hand, the built-time components refers to the design and definition of the workflow. Users define the tasks of the workflows, their dependencies, the platforms to use, etc. using the workflow language specification that will be detailed later. On the other hand, the run-time components comprises elements for the workflow execution and monitoring and others for the interaction with the computing/data resources. The workflow execution and control is managed

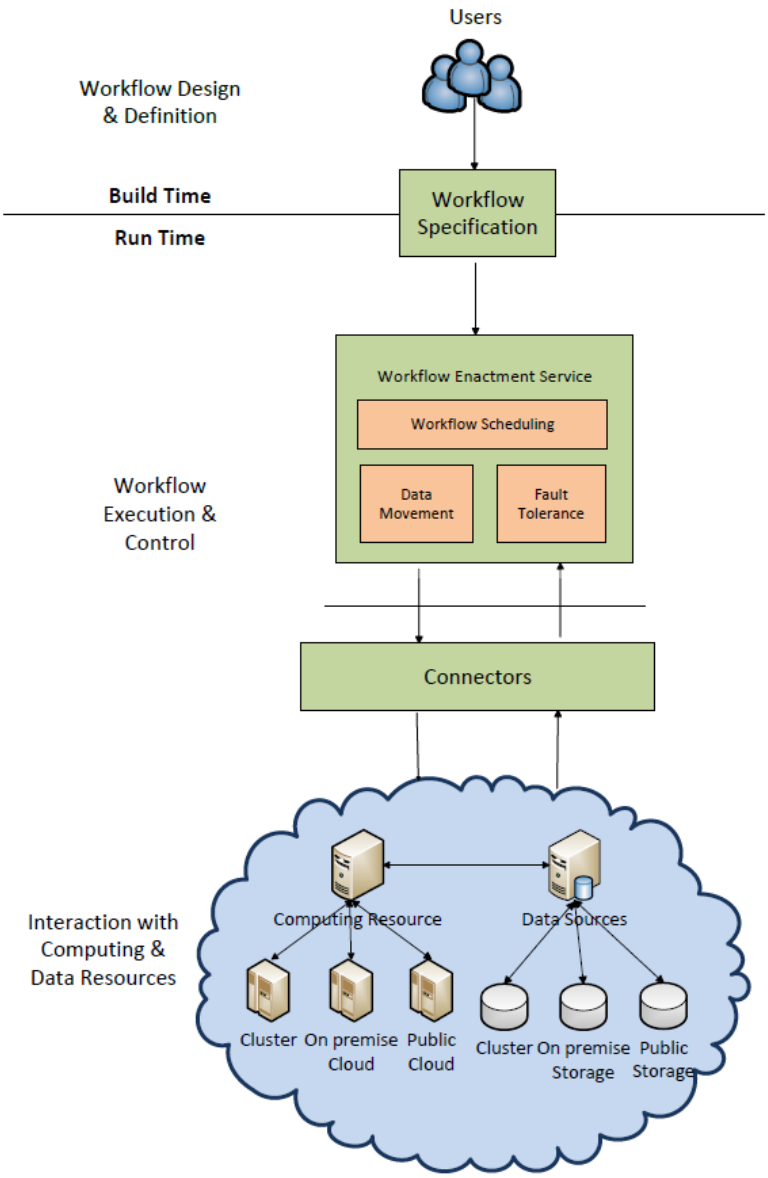


Figure 1: WMS architecture.

by the workflow enactment service. The most important functions of the enactor are scheduling the tasks to jobs, moving the data between resources and restoring the flow when a job fails. The interaction with the computing and data resources is implemented via connectors or drivers to the specific computing and data resources.

### 4.1. Design principles

Design principles represent a set of guidelines that avoid creating a bad architecture design. If the architecture adheres to the following principles, costs and maintenance will be minimized while usability and extendibility will be promoted. The

key principles of our architecture are:

• Platform-agnostic client. The client program has been developed using a platform-agnostic programming language and thus can be used in a wide spectrum of Operating Systems.

• Generality. It should be possible to execute any kind of workflow application that can be expressed as a Directed Acyclic Graph.

• Extensibility. The architecture can be extended to include new func- tionality such as support for a new computing and/or storage back- ends.

• Modularity. A change on a part of the system should not require changes on the rest of the system if the interfaces are preserved.

• Multi-platform. Each part of the workflow can be executed using different computing back-ends.

• NIST Cloud Computing definition compliant. When using cloud resources, the system follows the requirements expressed by the National Institute of Standards and Technology (NIST) cloud computing definition (see Definition 1).

Definition 1. “Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction”.

### 4.2. Workflow structure

Most scientific applications can be modelled using the workflow programming model. In this model, the application is composed of multiple tasks that are connected according to their dependencies. In our system workflows are expressed using Directed Acyclic Graphs (DAGs) where the nodes represent computational tasks and the directed edges the dependencies between them (see Figure 2). A task

has dependencies in the form of files and it will start its execution only when the output file(s) of the task(s) it depends on are available.

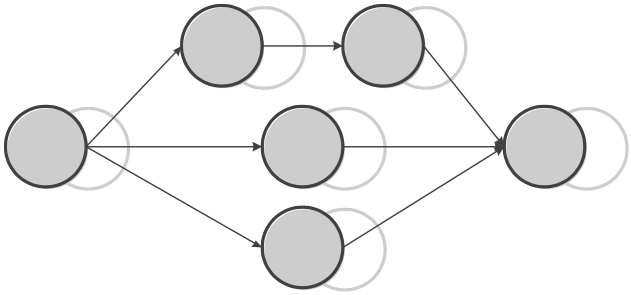

Figure 2: Aspect of a typical scientific application modelled as a DAG.

There are variations of the model in which the workflow also contains conditional branches (a task may be executed or not depending on the result of previous tasks) and loops (the execution of a part of the workflow is repeated). This workflow structure is known as non-DAG. Although there are WMSs that provide conditional and loop functionalities, the workflow language complexity is higher and therefore its adoption might be limited.

### 4.3. Workflow composition system

Workflow composition systems are designed to enable users to compose their workflows. For that purpose, systems must provide a high level view of the workflow applications, hiding the complex aspects of the underlying infrastructures. Following the taxonomy of workflow composition systems showed in [10], there are two types: user-directed and automatic. User directed composition systems allow users to edit workflows directly while automatic composition systems generate workflows for users automatically. In turn, user directed can be split into two categories: language-based modelling and graph-based modelling. Our WMS provides user-directed composition where users use language-based modeling.

While almost every state-of-the-art workflow uses markup languages, such as the XML format, for expressing workflows, we have chosen Java Script Object Notation (JSON) [19]. JSON offers some benefits over XML: it is less verbose, easier to write and read for humans and does not require writing end tags. Upon providing the JSON document, our system automatically validates the syntax and parses it using one of the many JSON processors available. Because the system is mostly

implemented in the Java programming language, we use Jackson [20] for parsing workflow documents.

### 4.4. Workflow specification

A workflow specification (also called workflow model) defines a workflow including its task definition and structure (task connectivity). There are two types of workflow models: abstract and concrete (executable).

#### 4.4.1. Abstract workflow

The abstract model describes the workflow in an abstract form without referring to the specific computing platforms for task execution. In fact, it is a template that includes the task that must be executed and for each of these tasks, the inputs, outputs, hardware requirements, commands and arguments to be used when invoked. The resource-agnostic nature of these descriptions provides a flexible way for users to define workflows without being concerned about low-level implementation details and thus, it can be ported to different computing infrastructures. In addition, abstract models facilitate the sharing of workflow descriptions between users working on the same field of interest. To exemplify this, Listing 1 shows a JSON template of a test workflow. The template is interpreted as follows: the workflow contains one stage named process0 that must invoked in the front-end node ramses using as Operating System the 64-bit version of Ubuntu. The desired hardware requirements are 4-single core nodes with 4GB RAM and a 20GB disk. The execution of the stage requires invoking, for each node, the program test using as arguments the filenames of two files contained in input0. The output of the stage (and in this case the output of the workflow) are all the .txt files generated by the program. Notice that all the values that start with '#' are references to JSON objects defined in the same file or the resource configuration file described below.

```
{
"stages": [
        {
                "id": "process0", "hostId": "#ramses", "environmentId":
        "#ubuntu64bit", "nodes": [
                {
                        "numNodes": "4", "coresPerNode": "1", "memorySize":
                "4096m", "disks": [
                        {
                                "nDisk": "0", "diskSize": "20g"
                        }
                                                        ]
                }
                                ],
        "execution": [
                {
                        "path": "./test", "arguments": "#input0(2)"
                }
                    ],
        "stageIn": [
                {
                        "id": "#input0"
                }
                   ],
        "stageOut": [
                {
                        "id": "output0", "type": "File", "filterIn": "*.txt", "replica":
                "none"
                }
                     ]
        }
          ]
}
```

Listing 1: Workflow template example

### 4.4.2. Resource information file

To transform the abstract workflow into a concrete or executable workflow, the WMS needs information about the hosts, the environments and the input files of the initial stages of the DAG (i.e. those stages that don't have input dependences with other stages). This information can be found in a JSON configuration file like the one showed in Listing 2 for the previous workflow. The array hosts contains a list with the data for accessing the front-end hosts, such as: the host name, type, port and different credentials depending on platform to be used (for instance, a certificate for Windows Azure and user/password pair for OpenNebula). Environments is an ad-hoc field for executions on cloud computing platforms. It defines the required features of

the VMI (Virtual Machine Image) to use as a base to create the VMs: Operating System and the software packages that should be installed on it. VMIs are obtained from the image repository associated to each de- ployment. Last, but not least, the section inputFiles declares the input files of the workflow: the identifier, the type (File, Parameter, etc.) and the physical location (URI).

```
{
"hosts": [
        {
                "hostId": "ramses", "type": "Cloud", "subType":
        "OpenNebula",
                "hostName": "ramses.i3m.upv.es", "port": "1111",
                "credentials": {
                        "userName": "userName", "password":
                "passWord"
                                }
        }
        ],
"environments": [
        {
                "environmentId": "ubuntu64bit", "osName": "linux",
                "arch": "x86_64", "osFlavour": "ubuntu", "osVersion":
        "14.04",
                "packages": [
                                "unzip"
                            ]
        }
                ],
"inputFiles": [
        {
                "id": "input0", "type": "File",
                 "values": [
                        "db.zip"
                            ],
                        "extract": "true"
        }
                ]
}
```

Listing 2: Configuration file

### 4.5. Workflow validation

Once the user has provided the abstract workflow specification and the resource configuration file to the system, both files are examined by a parser to

check if the syntax of these documents is compliant to the JSON specification. If the validation passes, the WMS performs a semantic validation of the JSON documents. With the intention of improving the usability of the system and avoid errors during the execution phase of the workflow, the system cross-validates the information provided in both documents. Among other aspects, the semantic validator checks that every reference (to a host, environment or input file) in the abstract workflow exists in the resource configuration file. Moreover, the identifiers of the different elements should be unique (two different stages cannot share the same id) and the values should match certain regular expressions (for instance, memory is an integer followed by the characters ‘m’(mega), ‘g’(giga) or ‘t’(tera)). If the semantic validator finds any error, it prompts to the user the erroneous file and line. Lastly, because the system only supports workflows that can be modelled as DAGs, the graph is explored to make sure that it does not contain any cycle.

### 4.6. Workflow planning

The build-time process of generating an executable workflow based on an abstract workflow is called workflow planning. During that process, the original workflow undergoes a series of refinements geared towards optimizing the overall performance and transformations for cloud-support and data management. Actually, the mapper process distinguishes our WMS from other systems by providing a novel approach that dynamically provisions and releases cloud computing resources.

As Figure 3 displays, the mapper performs three conversions to the abstract workflow. The first refinement (a) fuses two or more sequential tasks if the following conditions are given: all the stages are executed on the same infra- structure with the same environment, only the first stage is allowed to have input dependencies with other tasks and the last stage more than one output dependence. Of course, this conversion is not mandatory and its purpose is to optimize or simplify the execution flow. The second transformation (b) adds data management tasks (labelled as COPY) before every task of the workflow obtained in the previous step. The purpose of these tasks is to stage-in/out the required input by the tasks or output to the user selected location, respectively. Finally, if a task is executed on cloud computing resources, it is necessary to add tasks that deploy the resources only when they

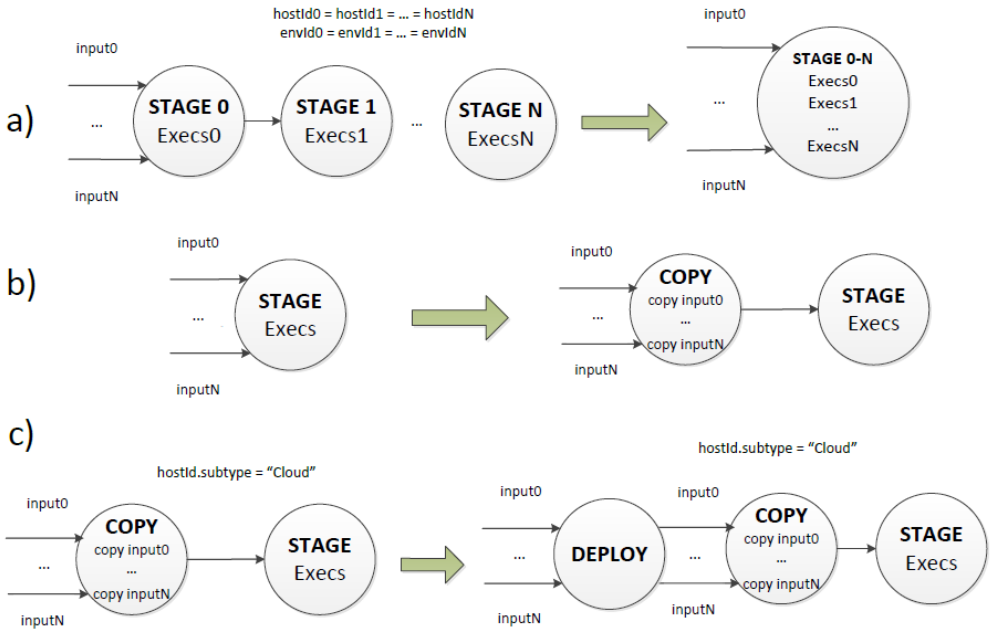


Figure 3: Planner conversions

are needed and undeploys them when the products have been copied to their destinations. In order to accomplish this, the planner adds (c) a synthetic task (called DEPLOY) before the stage-in task produced in (b) and another task (UNDEPLOY) after the stage-in of subsequent tasks.

Figure 4 shows the mapping process of a simple workflow with 4 sequential tasks where tasks A and B are executed on the same cloud platform with the same environment, C is executed on a cluster and D is also executed in a cloud infrastructure. Although the next subsection gives an extensive explanation of each task, the planner generates four different types of nodes: deploy, copy, undeploy, cleanup and copyout.

### 4.7. Workflow execution

Once the mapper has produced the executable, it is submitted to the workflow execution engine. The execution of the workflow begins with the initialization of every element: the state of the tasks are set to IDLE and the state of the inputs/outputs to DISABLED. Next, due to the data-flow nature

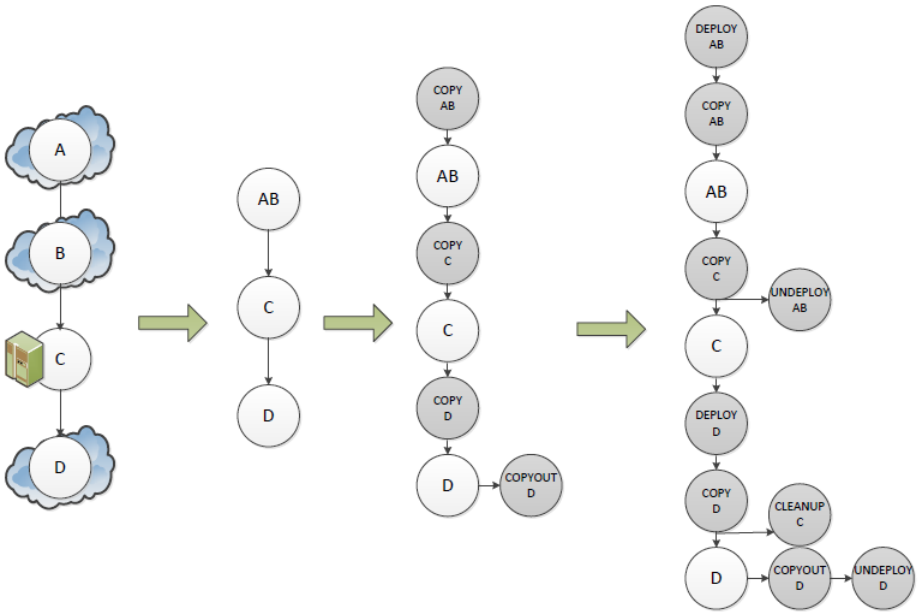


Figure 4: Translation of a simple abstract workflow to an executable workflow.

of the workflow system, the inputs provided by the user are ENABLED, allowing the execution of the first task(s). The pseudocode of the workflow execution engine is controlled by two core functions (see Figure 5): runTask and getStatus. The runtime checks if all the inputs of a task are enabled, calling runTask in that case. When a task is submitted, the engine periodic- ally monitors its status through the getStatus function and if it has finished successfully, enables the outputs of the tasks (which in turn are normally in- puts of the next tasks). Obviously, the behaviour of runTask and getStatus will vary according to infrastructure (cluster and cloud) and the task type (deploy, copy, user-defined, undeploy, cleanup or copyout).

### 4.7.1. Deploy task execution

The execution of a deploy task is required when the user desires to execute a task of the abstract workflow in a cloud platform. In order to dynamically deploy cloud computing resources, the system makes a request to the IM (Infrastructure Manager) [21]. The main function of the IM is to deploy and automatically configure the virtual infrastructure required to execute and manage an application in a cloud computing environment. The main features of this tool are:

```
1: Initialize listStage
2: while nTasks> 0 do
```

```
3:      for task= 0 to listStageSize  do
4:      taskStatus←getStatus()
5:      if taskStatus=ENABLED then
6:      taskStatus←RUNNING
7:      runTask()
8:      else
9:      if taskStatus=FINISHED then
10: nTasks=nTasks−1
11: listStage.drop(task)
12: task.enableOutputs()
13:      end if
14:      end if
15:      end for
16:      Sleep X minutes
17: end while
```

Figure 5: Workflow execution engine algorithm

• A language specification of software and hardware requirements for the user applications that can be used by both non-expert (since it is easy to encapsulate recipes as building blocks) and advanced users (due to its high expressivity), called RADL (Resource and Application Description Language).

• Another component, the VMRC (Virtual Machine Resource Catalog) [22] is used to select the most suitable Virtual Machine Image (VMI) based on the user expressed requirements.

• Provision of Virtual Machines on both, public clouds (Amazon EC2, Windows Azure, etc.) and private clouds (OpenNebula, OpenStack, etc.).

• Run-time contextualization of the infrastructure that installs and con- figures the software required that may not be pre-installed in the VMIs used.

• Elasticity management support.

• Last but not least, it provides two APIs to enable high-level components to access the functionality: XML-RPC and REST APIs. These APIs provide a set of simple functions for clients to create, destroy, and get information about the infrastructures.

The API used by the WMS is the one based on the XML-RPC protocol. On the one hand, the runTask function in a deploy task calls the deploy function of the API. Prior to it, the system needs to build a RADL document, using the hardware and software requirements of the task expressed in the JSON document. By means of a RADL document, the WMS calls the IM to configure the deployment as a Portable Batch System (PBS) cluster where all nodes share the same disk via NFS. In this manner, PBS acts as the scheduler of the jobs that the task should execute. On the other hand, the getStatus invokes the API function that queries the status of the infra-structure. The task is considered to be finished when the status returned by the API is configured. From this point on, the WMS interacts with the cloud infrastructure through SSH, using the information returned by the API (public IP and user credentials).

### 4.7.2. Copy task execution

The copy task is in charge of the data management during the execution, one of the most crucial parts of any WMS. Moreover, these tasks are executed regardless of the computing platform used (cluster or cloud). When runTask is called for a copy task, the first step is to declare an unique name for the execution directory (our system uses the current epoch time). Then, this execution identifier is used for creating the execution directory in the file system of the target infrastructure. Now that the execution directory is ready for hosting the task data, the function of runTask is staging-in the data. As a convention, our system distinguishes between two types of stage-ins: the ones that begin with the word input and the ones that begin with output. Inputs are user-provided data while outputs are data whose origin is another task of the workflow (i.e intermediate data).

With respect to the input data, the system can download any file that can be retrieved with the protocols supported by the unix wget command (http, https and ftp). If the URI of the input file defined in the configuration file does not use any of these protocols, the system assumes that the file is in the user machine where the workflow managament system resides (local file).

Another important aspect is the possibility of explicitly indicating that the input files should be extracted on the destination resources. However, since there are tools that require compressed data as input, this extraction should be optional. In any case, the stage-in of an input file triggers the submission of a job to the physical or

virtual cluster scheduler for downloading the file and next, if it is required, extracting the file. The system supports almost every common compression format (.zip, .rar, .gz, .tar).

The other type of stage-ins are the intermediate results produced by previous tasks in the DAG. To handle the transference of this kind of data, the WMS submits a basic job that invokes the scp (Secure Copy Protocol) program with the corresponding credentials and arguments.

The goal of getStatus in a copy task is to make sure that all the copy jobs submitted by runTask have finished successfully. If there is a least one job pending the status of the copy task returned is RUNNING, otherwise the system considers that the task is completed (status FINISHED) and enables the stage-outs of the task.

### 4.7.3. User-defined task execution

In contrast to the previous tasks, user-defined tasks are the same that appear in the abstract workflow specification but now they are executable. In our WMS, a user-defined task is said to be executable when two conditions are met: firstly, the target infrastructure is already available (the cluster is accessible or the cloud computing platform is deployed), and secondly, the input data needed by the tasks has been staged-in to these resources. As it can be appreciated, both conditions correspond to the actions performed by the deploy task and copy task, respectively.

According to the abstract workflow, a task can contain a block of executions or commands to execute. When the runTask function is invoked for this kind of tasks, the WMS analyses the commands to determine if there is parallelism in the submission of the job or not. The parallelism of a task is explicitly indicated by the user in the abstract workflow, appending the “(x)” expression to an argument where x is the granularity (i.e. the number of files used per job). For instance, let’s suppose the scenario showed in Figure 6. The WMS submits a job for each group of two files contained in db.zip. If after the analysis the token “(x)” is not found, then the system considers that the task is not parallel and only one job is submitted in that case.

However, the approach showed above is not efficient when dealing with

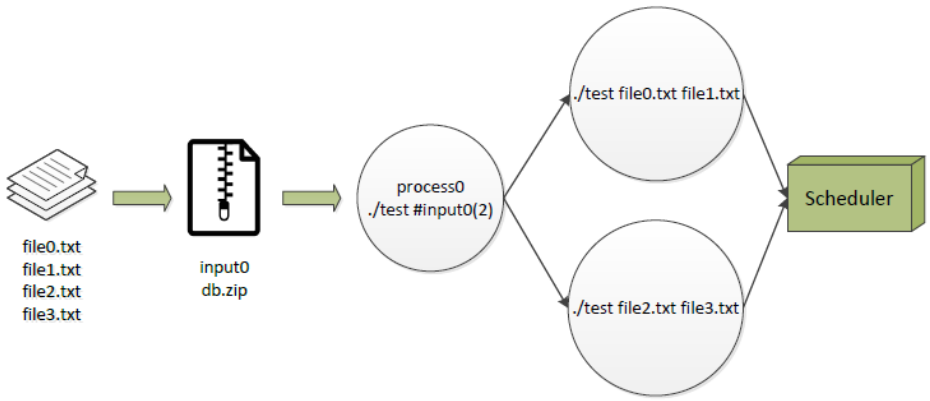


Figure 6: Execution of a parallel task.

short running tasks (on the order of minutes or seconds). One of the most common problems of distributed computing infrastructures is the overhead as a consequence of the queueing time on the computing resource schedulers. This fact results on an increase of the response time of the scientific applica- tions. In order to avoid this situation, the WMS implements task clustering techniques that group short tasks into coarse-grained tasks, thus greatly reducing the queuing time. Our WMS currently implements two clustering techniques, although, thanks to the modular design of the tool, advanced users can implement and include their own strategies with minimal effort. Random clustering. This strategy is recommended when the computa- tional cost of processing the input files is similar or unknown. The granular- ity of the cluster jobs (i.e. the number of tasks per clustered job) is controlled using two configuration parameters: wallTimePerJob (maximum execution time allowed for a clustered job) and wallTimePerTask (an estimation of the processing time of a single file or task). Using these two parameters, the system computes the clustering granularity, taking into account that: firstly, the number of jobs has to be greater or equal than the number of parallel instances available and secondly, the total estimated execution time of a single job cannot exceed the wallTimePerJob.

Size clustering. If the runtime of the tasks has a high variance, the previous technique may load balance poorly in some situations, producing clustered jobs of small tasks and others of larger tasks. In these cases, if the computational load of a task depends on the file size, the size clustering strategy can be used to create jobs with approximately the same total file size (i.e. the same amount of time required to process).

Once runTask has submitted all the jobs to the scheduler, the goal of get- Status is monitoring the status of all jobs until all of them reach a final state (finished

or failed). If the task executes many trivially parallel jobs, the enabling of the stage-outs can be done in two modes: economic and fast. The economic or standard mode is the one in which the getStatus function waits for every parallel job to have finished successfully before enabling the stage-outs. In the fast mode, getStatus can partially enable a stage-out each time that a parallel job has finished, allowing the deployment (in the case of the cloud), copy and maybe partial execution of the next tasks of the workflow. When using cloud computing infrastructures this behaviour can be very effective on reducing the response time of the scientific experiment but it also increases the usage of the infrastructures and its associated cost.

### 4.7.4. Undeploy task execution

Due to the variable demand of resources that scientific workflows exper- ience during the execution of the different stages, when a cloud computing task finishes and the output data has been staged-out, the resources assigned to it are no longer needed and they must be freed. This action follows the National Standards of Technology (NIST) definition of Cloud Computing which points out that resources should be "...released with minimal manage- ment effort or service provider interaction ". Moreover, because of the pay as you go model of this paradigm, the undeployment of resources keeps the user costs down. The dynamic provisioning and release of cloud computing resources is one of the main features that distinguishes our WMS from other tools with similar purposes.

As in the deployment task execution case, the runTask function calls the proper function of the IM XML-RPC API, destroyInfrastructure.

The aim of getStatus in this case is to make sure that the infrastructure re-moval operation is correctly carried out. This is especially important when public clouds are used to avoid incurring in unnecessary costs.

### 4.7.5. Cleanup task execution

The cleanup task is the equivalent of the undeploy task but for the case of clusters. Because a workflow task usually generates large of amounts of data and clusters are infrastructures shared with other users, a best practice consists on cleaning up the data once it has been staged-out. Thus, the function runTask simply deletes via SSH the whole execution directory created for the task and getStatus

makes sure that the operation is actually done.

### 4.7.6. Copyout task execution

From the user's point of view, the purpose of the copyout tasks is to retrieve the data products of the computations. The mapper attaches these special tasks only to the final tasks of the abstract workflow specification (i.e tasks which don't have dependencies with other tasks).

The runTask function starts the stage-out of the output to one or more locations. The default action is to transfer the data to the user local space (where the submit host is being executed). If besides the field replica of the output contains references to another data storage sites, the data will be also copied to these locations. The other function, getStatus, will monitor the data transference until all of them are completed.

## 4.8. Persistence

As it was mentioned before, we assume that the user has access to a machine where the WMS resides and it has permanent connection during the workflow execution. Nevertheless, a typical use case involves executing a scientific workflow composed of tasks with a elevated computational cost (in the order of days or even weeks) and so, demanding a permanent connection to the user machine is not a viable measure. For that reason, the system includes a persistence layer that periodically saves the state of the workflow, allowing users to interrupt the execution and resume it later. The persistence has been implemented using the NoSQL MongoDB [23]. In addition to the features offered by the NoSQL approach (simplicity of design, horizontal scaling, among others) over the traditional relational databases, MongoDB uses JSON-like documents, favouring the straightforward translation between the workflow descriptions and the database documents.

## 4.9. Fault tolerance

Although workflow execution failures in clusters and Cloud Computing platforms are not very common, fault tolerance is of most importance in distributed computing environments. It is necessary to provide the proper fault-tolerance

mechanisms to handle failures and support the reliable execu- tion in the presence of software or hardware failures. Due to the difference in terms of requirements between the tasks that compose a workflow, the WMS defines different fault tolerance policies for each task. The policies simply define the number of retries in case of software failure or hardware failure. The user also has the possibility of indicating such values in the abstract workflow specification, using the object retries and its fields OnWallTimeExceeded, OnSoftwareFailure and OnHardwareFailure inside a stage object. If, for some reason, a task exceeds the maximum number of retries for any type of failure, the execution of the workflow is automatically aborted.

### 4.10. Provenance

Workflow provenance is critical to users to be able to follow the evolution of their executions and to determine the cause behind a failure. In that sense, the WMS implements a logging system that registers in a file the events that occur during the execution along with other important information (for example, the elapsed time of a task that has finished). This information is not only useful for monitoring the progress of the experiment but also for getting performance statistics as we have done in the use case presented in the next section.

## 5. Use case: Orthosearch

The aim of this section is to describe the comparative genomics pipeline, Orthosearch, used as use case for testing the functionality of the WMS de- veloped in this work.

### 5.1. Preliminary concepts

Prior to describing the workflow, it is necessary to introduce the following concepts:

#### 5.1.1. Comparative genomics

Comparative genomics mainly refers to homology and evolutionary dy- namics between organisms, genes and proteins. Be it through complete or specific genomic

comparison, such discipline may provide a deeper understanding on how species evolved over time [24]. In addition, functional studies allow for a greater observation on health, phenotype, coding exons, noncoding RNAs and many other aspects related to the species genomic complexity and lineage-specific adaptations [25] [24].

### 5.1.2. Homology and homology inference

Homology is very broad comparative genomics concept, which comprises a relationship of common descent between genes or proteins. Even though there are several homology related scenarios, such as orthology, paralogy, horizontal gene transfer, gene loss, xenologs and others our work focuses on orthology and paralogy [25]. Orthology is characterized when the same genes or proteins are present in distinct species, due to a speciation event. Paralogy relates to duplicated genes - usually in the same species - although they may be inferred in distinct organisms [25].

As ortholog genes tend to preserve their ancestor function, these can be used in order to improve the annotation of data obtained from newly sequenced genes in several organisms. Furthermore, ortholog prediction can also be used to provide better understanding and evolutionary classification of such genes. [25] [26].

High quality ortholog prediction is a desirable aspect for many studies, especially when dealing with incomplete or lacking experimental genomic data [27]. In addition, it also has a direct impact on many comparative genomics tasks, such as functional characterization, genome annotation, conserved regulatory elements identification, orthologous databases creation and others [28] [29] [30] [31].

## 5.2. Orthosearch

OrthoSearch (Orthologous Gene Searcher) [32] [33] is a genomics comparative workflow. Initially conceived as a Perl-based routine, it is a profile-protein, reciprocal best hits (RBH) based solution for homology inference among species. It comprises several stages and uses distinct bioinformatics tools, such as Mafft [34] and HMMER [35] which confront an orthologous database with an organism multifasta protein data. The workflow structure is depicted in Figure 7.

OrthoSearch was initially built in order to infer homology between Protozoa species, although there is no restriction to any specific genome. It has already been

evaluated and proven to be effective when inferring orthology among five Protozoa organisms with two distinct orthologous databases [28], NCBIs COG and KOG [36].

## 6. Experimentation and results

### 6.1. Data selection

We selected a subset of EggNOG database version 4 [37] which comprises eukaryotic ortholog groups only EggNOG KOG. Its design comprises up- to-date techniques for ortholog groups construction, inparalogs recognition,

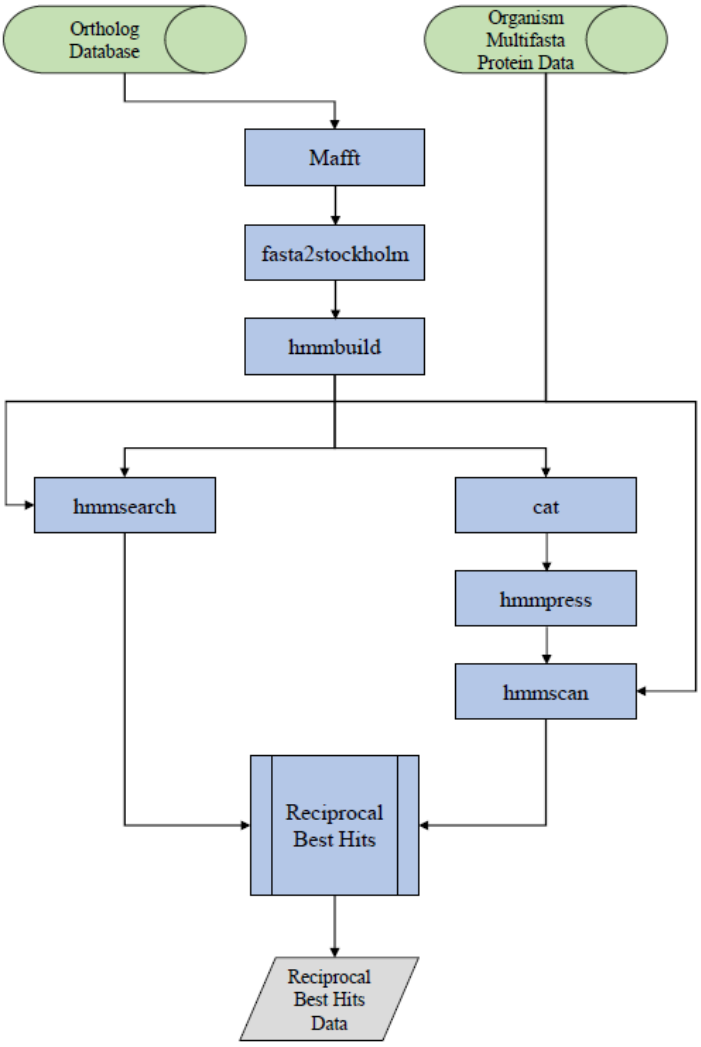


Figure 7: Orthosearch abstract workflow.

robust automatic annotation, single-copy ortholog groups nesting and re-

construction of phylogenetic trees based on the generated clusters multiple alignments.

Three Protozoa species were selected to be confronted with EggNOG KOG database: *Cryptosporidium hominis*, *Entamoeba histolytica* and *Leishmania* infantum. *Cryptosporidium* species causes acute gastroenteritis and diarrhea. It is potentially dangerous, with high levels of morbidity and mortality in AIDS patients [38]. There is no effective treatment or prevention for such infection in humans so far [39].

Amoebiasis is a disease caused by *Entamoeba histolytica*, a parasite of the human large intestine, ordinarily acquired by contaminated water or food ingestion. Its infection is usually associated to places with poor hygiene and sanitation conditions. Such disease leads to diarrhea and dysentery, causing over 55,000 yearly deaths [40] [41].

Visceral leishmaniasis is caused by *Leishmania infantum*. With an estimate

200,000 to 400,000 infections each year [42], it is an opportunistic and very dangerous infection for HIV patients as it promotes HIV clinical progression and reduces therapy response [43].

These Protozoa species are responsible for the death of thousands to millions humans. In addition, there are either no vaccines for such or the available treatments are mostly inadequate due to toxicity and drug resistance [44] [45]. Therefore, comparative genomics experiments among such pathogens genomes that may lead us to a deeper knowledge of these organisms biology are of public health interest. These may aid on the discovery of new issues related to the pathogenicity of such, as well as help to design new, more specific drugs to treat the infected patients or even prevent the infection itself.

### 6.2. Sequential execution

Figure 7 displays that the Orthosearch pipeline is composed of 8 stages or processes: Mafft, fasta2stockholm, hmmbuild, hmmsearch, cat, hmmpress, hmmscan, Reciprocal Best Hits.

#### 6.2.1. Computing resources configuration

Initially, the serialized version of the application was executed using a single Ubuntu 14.04 compute instance with 16 CPU cores, 24GB RAM and 100GB disk.

### 6.2.2. Performance results

Table 6.2.2 shows the elapsed time for every stage of Orthosearch and the three organisms confronted against the database.

| Stage/Organism | *C. hominis* | *E. histolytica* | *L. infantum* |
|---|---|---|---|
| **Mafft** | **1539 (37,8%)** | **1633 (33,93%)** | **1579 (31,7%)** |
| **fasta2stockholm** | 15 (0,36%) | 17 (0,35%) | 16 (0,32%) |
| **hmmbuild** | **1755 (43,20%)** | **1746 (36,28%)** | **1872 (37,58%)** |
| **hmmsearch** | 235 (5,78%) | 392 (8,14%) | 448 (8,99%) |
| **cat** | 46 (1,13%) | 53 (1,10%) | 63 (1,26%) |
| **hmmpress** | 49 (1,20%) | 49 (1,01%) | 53 (1,06%) |
| **hmmscan** | **404 (9,94%)** | **895 (18,6%)** | **922 (18,51%)** |
| **Best-Hits** | 19 (0,46%) | 27 (0,56%) | 28 (0,56%) |
| **Total Time** | 4062 | 4812 | 4981 |

Table 2: Elapsed time (minutes) for each stage using the sequential approach

In sight of the execution times obtained with the sequential pipeline, we identify three processes that take about 90% of the total time: Mafft, hmmbuild and hmmscan. Thus, when executing Orthosearch in distributed computing infrastructures the goal will be to reduce as much as possible the execution time of these phases.

## 6.3. Distributed computing execution

Before actually executing Orthosearch, the planner of the WMS was run for optimizing the global structure of the workflow. The result can be seen in the executable workflow (Figure 8): the 8 original stages of the pipeline have been grouped into the following three stages: Mafft/fasta2stockholm/hmmbuild (StageGroup1), hmmsearch, cat/hmmpress/hmmscan(StageGroup2) and Best- Hits.

### 6.3.1. Computing resources and execution configuration

After obtaining a preview of the executable workflow the next step was properly configuring the hardware resources and execution parameters according to the characteristics of each stage. Next, we show a synopsis (Table 3) that contains this information for each stage of the executable workflow.

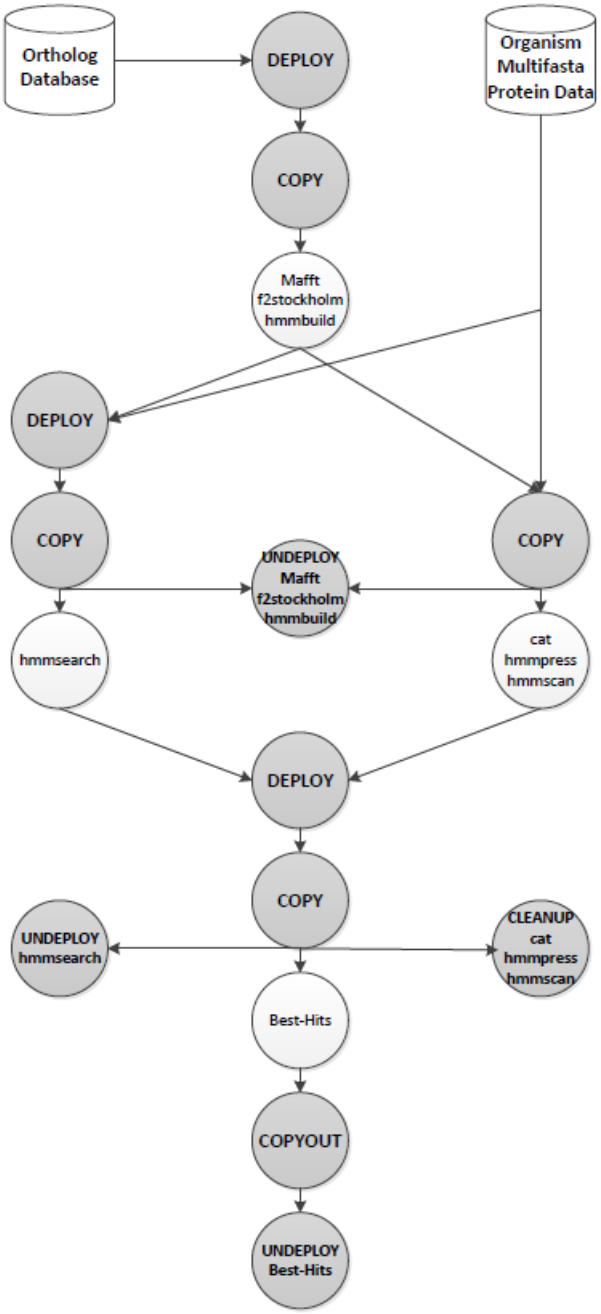


Figure 8: Executable workflow for Orthosearch.

| Stage | StageGroup1 | hmmsearch | StageGroup2 | Best-Hits |
|---|---|---|---|---|
| **Parallelism** | Trivially parallel | Trivially parallel | MPI | None |
| **Load balancing** | Size | Size | None | None |
| **Infrastructure type** | Cloud | Cloud | Cluster | Cloud |
| **Execution model** | Fast | Fast | None | None |
| **Number of nodes** | 16 | 16 | 6 | 1 |
| **Cores per node** | 1 | 1 | 1 | 1 |
| **Memory per node** | 4GB | 28 4GB | 32GB | 24GB |
| **Disk size** | 40GB NFS | 40GB NFS | 500GB | 50GB |

Table 3: Hardware configuration and execution parameters

According to the information showed in the Table, StageGroup1 and hmmsearch have a trivially parallel nature and so fit very well the computational model of cloud computing assets. Moreover, because the computing cost of the executions depends on the file size, the load balancing strategy selected has been by size. These two stages were executed each on an on premise cloud infrastructure managed by OpenNebula 4.8 cloud middleware with a virtual cluster of 16 Ubuntu 14.04 single-core VMs, sharing a 40GB disk via NFS. Due to the free-cost of running these VMs the WMS was con- figured for running on fast mode (enable the stage-outs as soon as the first parallel job finishes). On the other hand, the stage cat/hmmpress/hmmscan contains a task that can be speed-up only using a specific parallel version implemented on MPI, MPI-hmmscan. Thus, the infrastructure chosen as optimal for this stage was a cluster (kahan.dsic.upv.es) with Infiniband interconnection network. Finally, the stage Best-Hits does not admit any kind of parallelism and thus it was executed on a similar cloud deployment as the one used for Mafft/fasta2stockholm/hmmbuild and hmmsearch.

### 6.3.2. Performance results

Table 4 shows the performance results obtained when deploying the Orthosearch workflow with the configuration mentioned above.

| | *C. hominis* | *E. histolytica* | *L. infantum* |
|---|---|---|---|
| **Total time** | 414 minutes | 611 minutes | 575 minutes |
| **Speed-up** | 5X | 3,84X | 4,13X |

Table 4: Response time and speed-up ratio for each scenario.

In our best scenario (*Cryptosporidium Hominis*), the WMS provided a 5.0

speedup ratio, with a total 6 hours and 54 minutes of execution time. The same experiment, in a sequential approach, required 34 hours and 10 minutes. Our worst-case scenario, *Entamoeaba Histolytica* was also able to provide a remarkable 3.84 speedup. The average speed-up ratio for the three scenarios was: 4.32.

## 7. Conclusions and future directions

In the context of e-Science, the effective and efficient use of all the available computational resources is becoming increasingly important for performing scientific research, due to the overflowing amount of data being generated. Because e-Science processes are modelled with workflows, software components called Workflow Management Systems (WMSs) play a crucial role in this data deluge scenario.

Moreover, the advent of Cloud Computing and its core characteristics (rapid elasticity, resource pooling, and pay-per-use, among others) are well-suited to the nature of scientific applications that experience a variable demand during its execution. As a consequence, many WMSs derived from projects in the area of grid computing were updated to support the execution on Cloud resources. However, many of their features are optimized for grids and thus are unable to obtain the most key aspects of clouds, such as dynamic provisioning of resources. For that reason, in this work we present a novel multi-platform (clusters and clouds) WMS with support for on-demand provisioning of multi-cloud (public, private and hybrid) customized cloud computing resources.

The tool developed in this work has been tested using a comparative genomics pipeline, called Orthosearch. Promising results were obtained, with significant speedup ratio compared to the batch-oriented pipeline.

The current working lines include adding new features and improving others. In the first group the priority is to add support for Grid infrastructures and to make possible the interoperability of these platforms with clusters, supercomputers and any cloud that can be deployed with the Infrastructure Manager. On the other side, we want to replace SSH with a more efficient transference protocol, offer data privacy and last but not least, to reuse VMs between different stages through horizontal (adding and removing VM instances) and vertical (increasing and decreasing VM capacity) elasticity.

**Acknowledgments**

This paper wants to acknowledge the support of the EUBrazilCC project, funded by the European Commission (STREP 614048) and the Brazilian MCT/CNPq N◦ 13/2012, for the use of its infrastructure. The authors would like also to thank the Spanish “Ministerio de Economía y Competitividad” for the project “Clusters Virtuales El´asticos y Migrables sobre Infraestructuras Cloud H´ıbridas” with reference TIN2013-44390-R.

**References**

[1] D. Atkins, Revolutionizing science and engineering through cyberinfrastructure: Report of the national science foundation blue-ribbon advis- ory panel on cyberinfrastructure.

[2] S. Bohle, What is e-science and how should it be managed?, Nature.com, Spektrum der Wissenschaft (Scientific American), http://www.scilogs.com/scientificandmedicallibraries/what-is-e-science-and-how-should-it-be-managed.

[3] E. Deelman, D. Gannon, M. Shields, I. Taylor, Workflows and e-science: An overview of workflow system features and capabilities, Future Generation Computer Systems 25 (5) (2009) 528–540.

[4] C. Hoffa, G. Mehta, T. Freeman, E. Deelman, K. Keahey, B. Berriman, J. Good, On the Use of Cloud Computing for Scientific Workflows, in: 2008 IEEE Fourth International Conference on eScience, IEEE, 2008, pp. 640–645. doi:10.1109/eScience.2008.167. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4736878

[5] R. N.CalHeiros, R. Buyya, Adaptive execution of scientific workflow application on clouds, in: O. Terzo, L. Mossucca (Eds.), Cloud Computing with e-Science Applications, CRC Press, NW, 2015.

[6] M. Armbrust, O. Fox, R. Griffith, A. D. Joseph, Y. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, et al., M.: Above the clouds: a berkeley view of cloud computing.

[7] R. Figueiredo, P. Dinda, J. Fortes, A case for grid computing on virtual machines, in: 23rd International Conference on Distributed Computing Systems, 2003. Proceedings., IEEE, 2003, pp. 550–559. doi:10.1109/ICDCS.2003.1203506. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1203506

[8] W. Huang, J. Liu, B. Abali, D. K. Panda, A case for high performance computing with virtual machines, in: Proceedings of the 20th annual international conference on Supercomputing - ICS ’06, ACM Press, New York, New York, USA, 2006, p. 125. doi:10.1145/1183401.1183421. URL http://dl.acm.org/citation.cfm?id=1183401.1183421

[9] I. J. Taylor, E. Deelman, D. B. Gannon, M. Shields, Workflows for e-Science: scientific workflows for grids, Springer Publishing Company, Incorporated, 2014.

[10] J. Yu, R. Buyya, A Taxonomy of Workflow Management Systems for Grid Computing, Journal of Grid Computing 3 (3-4) (2006) 171–200. doi:10.1007/s10723-005-9010-8. URL http://link.springer.com/10.1007/s10723-005-9010-8

[11] E. Deelman, K. Vahi, G. Juve, M. Rynge, S. Callaghan, P. J. Maechling, R. Mayani, W. Chen, R. Ferreira da Silva, M. Livny, K. Wenger, Pegasus, a workflow management system for science automation, Future Generation Computer Systems 46 (2015) 17–35. doi:10.1016/j.future.2014.10.008. URL http://www.sciencedirect.com/science/article/pii/ S0167739X14002015

[12] K. Wolstencroft, R. Haines, D. Fellows, A. Williams, D. Withers, S. Owen, S. Soiland-Reyes, I. Dunlop, A. Nenadic, P. Fisher, J. Bhagat, K. Belhajjame, F. Bacall, A. Hardisty, A. Nieva de la Hidalga, M. P. Balcazar Vargas, S. Sufi, C. Goble, The Taverna workflow suite: designing and executing workflows of Web Services on the desktop, web or in the cloud., Nucleic acids research 41 (Web Server issue) (2013) W557–61. doi:10.1093/nar/gkt328. URL http://nar.oxfordjournals.org/content/early/2013/05/02/nar.gkt328.short

[13] C. A. Goble, J. Bhagat, S. Aleksejevs, D. Cruickshank, D. Michaelides, D. Newman, M. Borkum, S. Bechhofer, M. Roos, P. Li, D. De Roure, myExperiment: a repository and social network for the sharing of bioinformatics workflows., Nucleic acids research 38 (Web Server issue) (2010) W677–82. doi:10.1093/nar/gkq429. URL http://nar.oxfordjournals.org/content/38/suppl_2/ W677.short

[14] I. Altintas, C. Berkley, E. Jaeger, M. Jones, B. Ludascher, S. Mock, Kepler: an extensible system for design and execution of scientific workflows, in: Proceedings. 16th International Conference on Scientific and Statistical Database Management, 2004., IEEE, 2004, pp. 423–424. doi:10.1109/SSDM.2004.1311241. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1311241

[15] The ptolemy project, http://ptolemy.eecs.berkeley.edu/, accessed: 2015-02-24.

[16] J. Goecks, A. Nekrutenko, J. Taylor, Galaxy: a comprehensive approach for supporting accessible, reproducible, and transparent computational research in the life sciences., Genome biology 11 (2010) R86. doi:10.1186/gb-2010-11-8-r86.

[17] M. Albrecht, P. Donnelly, P. Bui, D. Thain, Makeflow: A portable abstraction for data intensive computing on clusters, clouds, and grids, in: Proceedings of the 1st ACM SIGMOD Workshop on Scalable Workflow

Execution Engines and Technologies, ACM, 2012, p. 1.

[18] R. Buyya, S. Pandey, C. Vecchiola, Cloudbus toolkit for market-oriented cloud computing, Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 5931 LNCS (2009) 24–44. arXiv:0910.1974, doi:10.1007/978-3-642-10665-1 4.

[19] Javascript object notation, http://json.org/, accessed: 2015-02-24.

[20] Jackson:high-performance json processor, http://jackson.codehaus.org/, accessed: 2015-02-24.

[21] M. Caballer, I. Blanquer, G. Moltó, C. de Alfonso, Dynamic Management of Virtual Infrastructures, Journal of Grid Computingdoi:10.1007/s10723-014-9296-5. URL http://link.springer.com/10.1007/s10723-014-9296-5

[22] J. V. Carrión, G. Moltó, C. De Alfonso, M. Caballer, V. Hernáandez, A Generic Catalog and Repository Service for Virtual Machine Images, 2nd International ICST Conference on Cloud Computing CloudComp2010 (Vm).

[23] Mongodb, http://www.mongodb.org/, accessed: 2015-02-24.

[24] R. C. Hardison, Comparative genomics., PLoS biology 1 (2) (2003) E58. doi:10.1371/journal.pbio.0000058. URL http://journals.plos.org/plosbiology/article?id=10.1371/journal.pbio.0000058

[25] E. V. Koonin, Orthologs, paralogs, and evolutionary genomics., Annual review of genetics 39 (2005) 309–38. doi:10.1146/annurev.genet.39.073003.114725. URL http://www.annualreviews.org/doi/abs/10.1146/annurev.genet.39.073003.114725

[26] E. V. Koonin, M. Y. Galperin, Sequence - Evolution - Function (2003). URL http://www.ncbi.nlm.nih.gov/books/NBK20260/

[27] C. Dessimoz, T. Gabald´on, D. S. Roos, E. L. L. Sonnhammer, J. Herrero, Toward community standards in the quest for orthologs., Bioinformatics (Oxford, England) 28 (6) (2012) 900–4. doi:10.1093/bioinformatics/bts050. URL http://bioinformatics.oxfordjournals.org/content/28/6/900.short

[28] R. R. C. Cuadrat, S. M. da Serra Cruz, D. A. Tschoeke, E. Silva, F. Tosta, H. Juc´a, R. Jardim, M. L. M. Campos, M. Mattoso, A. M. R. D´avila, An orthology-based analysis of pathogenic protozoa impacting global health: an improved comparative genomics approach with prokaryotes and model eukaryote orthologs., Omics : a journal of integrative

biology 18 (8) (2014) 524–38. doi:10.1089/omi.2013.0172. URL http://online.liebertpub.com/doi/abs/10.1089/omi.2013.0172

[29] D. L. Fulton, Y. Y. Li, M. R. Laird, B. G. S. Horsman, F. M. Roche, F. S. L. Brinkman, Improving the specificity of high-throughput ortholog prediction., BMC bioinformatics 7 (1) (2006) 270. doi:10.1186/1471-2105-7-270. URL http://www.biomedcentral.com/1471-2105/7/270

[30] C. Dessimoz, Editorial: Orthology and applications., Briefings in bioinformatics 12 (5) (2011) 375–6. doi:10.1093/bib/bbr057. URL http://bib.oxfordjournals.org/content/12/5/375.short

[31] L. Li, C. J. Stoeckert, D. S. Roos, OrthoMCL: identification of ortholog groups for eukaryotic genomes., Genome research 13 (9) (2003) 2178–89. doi:10.1101/gr.1224503. URL http://genome.cshlp.org/content/13/9/2178.short

[32] S. M. S. da Cruz, M. Mattoso, V. Batista, A. M. R. D´avila, E. Silva, F. Tosta, C. Vilela, M. L. M. Campos, R. Cuadrat, D. Tschoeke, OrthoSearch, in: Proceedings of the 2008 ACM symposium on Applied computing - SAC '08, ACM Press, New York, New York, USA, 2008, p.1282. doi:10.1145/1363686.1363983. URL http://dl.acm.org/citation.cfm?id=1363686.1363983

[33] S. M. S. da Cruz, V. Batista, E. Silva, F. Tosta, C. Vilela, R. Cuadrat, D. Tschoeke, A. M. R. D´avila, M. L. M. Campos, M. Mattoso, Detecting distant homologies on protozoans metabolic pathways using scientific workflows., International journal of data mining and bioinformatics 4 (3) (2010) 256–80. URL http://www.ncbi.nlm.nih.gov/pubmed/20681479

[34] K. Katoh, D. M. Standley, MAFFT multiple sequence alignment software version 7: improvements in performance and usability., Molecular biology and evolution 30 (4) (2013) 772–80. doi:10.1093/molbev/mst010. URL http://mbe.oxfordjournals.org/content/30/4/772.short

[35] R. D. Finn, J. Clements, S. R. Eddy, HMMER web server: interactive sequence similarity searching., Nucleic acids research 39 (Web Server issue) (2011) W29–37. doi:10.1093/nar/gkr367. URL http://nar.oxfordjournals.org/content/early/2011/05/18/nar.gkr367.short

[36] R. L. Tatusov, N. D. Fedorova, J. D. Jackson, A. R. Jacobs, B. Kiryutin, E. V. Koonin, D. M. Krylov, R. Mazumder, S. L. Mekhedov, A. N. Nikolskaya, B. S. Rao, S. Smirnov, A. V. Sverdlov, S. Vasudevan, Y. I. Wolf, J. J. Yin, D. a. Natale, The COG database: an updated version includes eukaryotes., BMC bioinformatics 4 (2003) 41. doi:10.1186/1471-2105-4-41.

[37] S. Powell, K. Forslund, D. Szklarczyk, K. Trachana, A. Roth, J. Huerta-Cepas, T. Gabald´on, T. Rattei, C. Creevey, M. Kuhn, L. J. Jensen, C. von Mering, P. Bork, eggNOG v4.0: nested orthology inference across

3686 organisms., Nucleic acids research 42 (Database issue) (2014) D231–9. doi:10.1093/nar/gkt1253. URL http://nar.oxfordjournals.org/content/early/2013/11/30/nar.gkt1253.short

[38] M. S. Abrahamsen, T. J. Templeton, S. Enomoto, J. E. Abrahante, G. Zhu, C. A. Lancto, M. Deng, C. Liu, G. Widmer, S. Tzipori, G. A. Buck, P. Xu, A. T. Bankier, P. H. Dear, B. A. Konfortov, H. F. Spriggs, L. Iyer, V. Anantharaman, L. Aravind, V. Kapur, Complete genome sequence of the apicomplexan, Cryptosporidium parvum., Science (New York, N.Y.) 304 (5669) (2004) 441–5. doi:10.1126/science.1094786. URL http://www.sciencemag.org/content/304/5669/441.short

[39] P. Xu, G. Widmer, Y. Wang, L. S. Ozaki, J. M. Alves, M. G. Ser- rano, D. Puiu, P. Manque, D. Akiyoshi, A. J. Mackey, W. R. Pear- son, P. H. Dear, A. T. Bankier, D. L. Peterson, M. S. Abrahamsen, V. Kapur, S. Tzipori, G. A. Buck, The genome of Cryptosporidium hominis., Nature 431 (7012) (2004) 1107–12. doi:10.1038/nature02977. URL http://dx.doi.org/10.1038/nature02977

[40] H. A. Lorenzi, D. Puiu, J. R. Miller, L. M. Brinkac, P. Amedeo, N. Hall, E. V. Caler, New assembly, reannotation and analysis of the Entamoeba histolytica genome reveal new genomic features and protein content information., PLoS neglected tropical diseases 4 (6) (2010) e716. doi:10.1371/journal.pntd.0000716. URL http://journals.plos.org/plosntds/article?id=10.1371/journal.pntd.0000716\#pntd-0000716-g004

[41] R. Lozano, M. Naghavi, K. Foreman, Global and regional mortality from 235 causes of death for 20 age groups in 1990 and 2010: a sys- tematic analysis for the Global Burden of Disease Study 2010., Lancet 380 (9859) (2012) 2095–128. doi:10.1016/S0140-6736(12)61728-0. URL http://www.sciencedirect.com/science/article/pii/ S0140673612617280

[42] J. Alvar, I. D. V´elez, C. Bern, M. Herrero, P. Desjeux, J. Cano, J. Jannin, M. den Boer, Leishmaniasis worldwide and global estimates of its incid- ence., PloS one 7 (5) (2012) e35671. doi:10.1371/journal.pone.0035671. URL http://journals.plos.org/plosone/article?id=10.1371/journal.pone.0035671\#pone-0035671-t014

[43] J. Alvar, P. Aparicio, A. Aseffa, M. Den Boer, C. Can˜avate, J.-P. Dedet, L. Gradoni, R. Ter Horst, R. L´opez-V´elez, J. Moreno, The relationship between leishmaniasis and AIDS: the second 10 years., Clinical microbiology reviews 21 (2) (2008) 334–59, table of contents. doi:10.1128/CMR.00061-07. URL

[44] M. P. Barrett, S. L Croft, Management of trypanosomiasis and leishmaniasis., British medical bulletin 104 (2012) 175–96. doi:10.1093/bmb/lds031.URL http://bmb.oxfordjournals.org/content/early/2012/11/22/bmb.lds031.short

[45] W. Gatei, C. N. Wamae, C. Mbae, A. Waruru, E. Mulinge, T. Waithera, S. M. Gatika, S. K. Kamwati, G. Revathi, C. A. Hart, Cryptosporidiosis: prevalence, genotype analysis, and symptoms associated with infections in children in Kenya, Am J Trop Med Hyg 75 (1) (2006) 78–82. URL http://www.ajtmh.org/content/75/1/78.short

## 8.5 Anexo E – Artigo 5 – "A Multi-Platform Workflow Management System optimized for Cloud Computing Platforms"

Carrión A, Caballer M, Blanquer I, **Kotowski N**, Dávila AMR. **"A Multi-Platform Workflow Management System optimized for Cloud Computing Platforms".**

**Este artigo foi publicado na conferência "21st International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA'15)".**

***A Multi-Platform Workflow Management System optimized for Cloud Computing Platforms***

Autores: Carrión, A.; Caballer, M.; Blanquer, I.; Kotowski, N.; Dávila, A.M.R

Identificador do manuscrito: PDP3232

Associado ao objetivo específico 2.2.3, este artigo apresenta um caso de uso, na forma de experimento, do elastic-OrthoSearch em plataforma computacional híbrida, composta por uma nuvem e por um *cluster*. Foi desenvolvido no I3M/UPV, está relacionado à tese, mas não compõe parte do escopo original da mesma, sendo a análise de tal cenário interesse principal do I3M/UPV.

# A Multi-Platform Workflow Management System optimized for Cloud Computing Platforms

**A. Carrión[1], M. Caballer[1], I. Blanquer[1], N. Kotowski[2] and A.M.R. Dávila[2]**
[1]Instituto de Instrumentación para Imagen Molecular (I3M),
Centro mixto CSIC - Universitat Politècnica de València - CIEMAT, Valencia, España
[2]Instituto Oswaldo Cruz (IOC), Fundação Oswaldo Cruz (FIOCRUZ),
Rio de Janeiro, Brazil

**Abstract**— *The scientific experimentation is facing a data deluge in which the amount of data generated is reaching the order of terabytes per day, and thus huge capacity is required to process this data. Computationally, these processes are modelled using Scientific Workflows. However, the execution of a Scientific Workflow can be a complex and resource-demanding task that must be managed by Workflow Management Systems (WMSs). As new computing paradigms emerge and infrastructures evolve, WMSs are extended to support these new computing back-ends. In fact, in the last years, Cloud Computing has appeared as another viable platform for running scientific applications. However, current WMSs are not optimized to exploit key cloud features. For that reason, this work details the design and implementation of a multi-platform WMS with a novel approach for efficiently supporting cloud resources. The engine has been successfully tested in the execution of a comparative genomics workflow called Orthosearch.*

**Keywords:** Workflow, Workflow Management Systems, Cloud Computing, Comparative-genomics.

## 1. Introduction

The relation between Science and computing goes back to the 1960s, when powerful computers (supercomputers) were introduced for performing scientific and engineering problems. At that time, a typical experimental scenario consisted in a repetitive cycle of moving data to a supercomputer for processing, submitting the executions and retrieving the outputs from the data storage [1]. Obviously, this process had to be automated for allowing scientists to focus on their research and not in the computational management. Fortunately, at the same time, the business community was addressing how to automate business processes and as a result the *Workflow* concept was born. In the business context, a *Workflow* can be defined as the orchestration of a set of activities in order to accomplish a larger and sophisticated goal. A specialization of this idea was adopted by the research community to model e-Science processes, the Scientific Workflows (SWFs). In this programming model, scientific applications are described as a set of tasks that have dependencies between them. It means that a task will start its execution only when the tasks it depends on have completed their execution.

The execution of workflow applications is a task with many details. A typical workflow is composed of hundreds of tasks that must be executed in a coordinated way. Moreover, all these tasks must be submitted to specific computing resources and the required inputs must be made available to the application. Software in charge of dealing with all these aspects are called Workflow Management Systems.

As new computing paradigms emerge and infrastructure evolve, so do the WMSs that support these computing back-ends. Traditionally, Scientific Workflow Applications have been extensively deployed in high-performance computing infrastructures, such as powerful clusters and supercomputers. Later, a highly distributed infrastructure, the Grid, appeared as an alternative to traditional approaches. In the last years, a new distributed computing paradigm, Cloud Computing, has appeared as another viable [2] platform for running scientific applications. In fact, some of their main features, such as rapid elasticity, resource pooling, and pay per use, are well suited to the nature of scientific applications that experience a variable demand during its execution.

Because the scientific experimentation is suffering from a data deluge phenomenon where the amount of data generated is reaching the order of terabytes per day, a huge amount of resources are needed to process this data and enable research. For that reason, it is crucial that WMSs have multi-platform support. Although current WMSs already support various platforms for the execution of workflow applications, many desirable features of cloud computing are not implemented, such as the dynamic provisioning of resources. For that reason, in this work we present a WMS with multi-platform support but also optimized to execute workflow applications in cloud computing platforms.

The remainder of the paper is structured as follows. Firstly, Section 2 gives an overview of the state-of-the-art WMSs. Afterwards, Section 3 explains in detail the architecture of a novel multi-platform WMS with actual support of cloud computing resources. Section 4 describes the comparative-genomics workflow, Orthosearch, selected as use case and Section 5 the experimentation and results obtained with it. Finally, conclusions and future working lines are exposed.

## 2. Related work

Due to the crucial role that workflow applications play in the scientific community, most current WMSs were developed to enable the execution of these applications in grid computing platforms. When Clouds became mainstream, WMSs were enhanced to support it. In this section, we present a brief description of the most prominent WMS found in the state-of-the-art which are related with our work.
Pegasus [3] is a mature Workflow Management System that combines features such as portability across a wide range of infrastructures, scalability, data management capabilities, exhaustive monitoring and complex workflow restructuring or transformations. It can be used with popular programming languages among the scientific community (such as Java, Python Perl) through its APIs (application programming interfaces) and also supports submission via web portals. Although it supports multiple cloud providers, it does not dynamic provision resources with different hardware and software requirements.
Taverna [4] is a WMS with a strong focus on bioinformatics where all computational workflow steps are Web Services. Workflows can be designed and executed on local desktop machines through the workbench or through other clients or web interfaces using the server mode. The server supports requests from many users to execute remote workflows with support of both grid and cloud platforms. It uses myExperiment [5] as repository for sharing and reusing workflows and the BioCatalogue and Biodiversity catalogue for Web Services discovery.
Kepler [6] is build upon the mature, dataflow-oriented Ptolemy system. From it, Kepler inherits several features such as the GUI and the Actor-Director model. In Kepler, workflows can be created connecting components called Actors which will process the data. An actor has several Ports for expressing input and output data and the director determines the model of computation used by the workflow. To the best of our knowledge, it only supports distributed execution via Web and Grid services, but not Cloud Computing platforms.
Galaxy [7] is an open, web-based approach that facilitates genomics research. It provides a collaborative environment for performing complex analyses, with automatic provenance tracking, allowing the transparent sharing of computational details, intent and context. Its objective is to offer accessible, reproducible and transparent computational research. A Galaxy instance supports running on compute clusters through Portable Batch System (PBS) and Sun Grid Engine (SGE).
The WMS described in this work distinguishes from the previous ones with features such as multi-cloud (public, private, hybrid) and multi-platform support (clusters and clouds). In addition, it presents a novel approach for the on-demand provisioning of cloud resources with ad-hoc hardware and software requirements. The solution will be released under GPL v3 license and it will be available for downloading at github (https://github.com/abel-carrion) very soon.

## 3. System architecture

The aim of this section is to describe the design and implementation of the architecture behind the WMS developed. The overall organization of the system is depicted in Figure 1. This schema is based on the one showed in [8], one of the most cited papers about the taxonomy of Grid WMSs. Our architecture is almost identical but extended to support a multi-platform scenario. In fact, our WMS currently supports execution on clusters, public clouds (Amazon EC2, Google Cloud Platform and Microsoft Azure) and private clouds (OpenNebula and OpenStack).

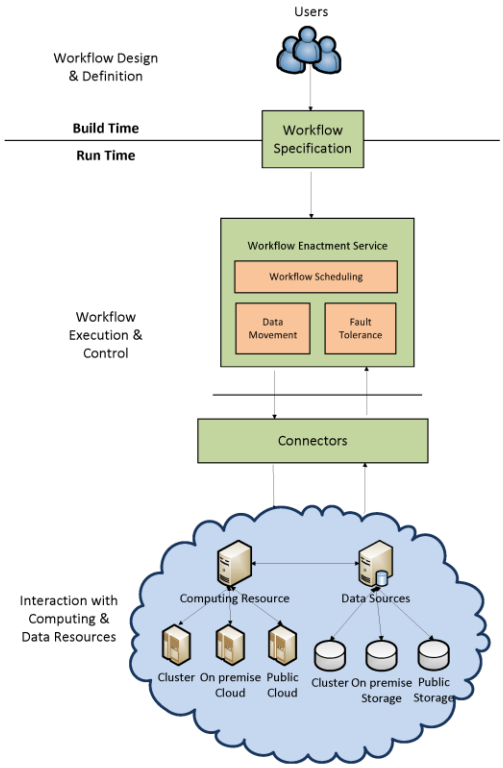


Fig. 1: WMS architecture.

### 3.1 Design principles

The key principles of our architecture are:

- **Platform-agnostic client**. The client program has been developed using a platform-agnostic programming lan-

guage and thus can be used in a wide spectrum of Operating Systems.
- **Generality**. It should be possible to execute any kind of workflow application that can be expressed as a Directed Acyclic Graph.
- **Extensibility**. The architecture can be extended to include new functionality such as support for a new computing and/or storage back-ends.
- **Modularity**. A change on a part of the system should not require changes on the rest of the system if the interfaces are preserved.
- **Multi-platform**. Each part of the workflow can be executed using different computing back-ends.
- **NIST Cloud Computing definition compliant**. When using cloud resources, the system follows the requirements expressed by the National Institute of Standards and Technology (NIST) cloud computing definition.

## 3.2 Workflow structure

In our system, workflows applications are composed of a number of tasks which have data dependencies (in the form of files) between them. A task depends on the output(file(s)) of one or more tasks to be used as its input. Only when the inputs of the task are available, it will start its execution.
In formal terms, these workflows can be represented by a Directed Acyclic Graph (DAG) where the nodes represent computational tasks and the directed edges the dependencies between them.

## 3.3 Workflow specification

A workflow specification (also called workflow model) defines a workflow including its task definition and structure (task connectivity). There are two types of workflow models: abstract and concrete (executable).

### 3.3.1 Abstract workflow

The abstract workflow specification is a template that describes the tasks that must be executed and for each of these tasks, the inputs, outputs, commands and arguments to be used when invoked. The resources-independent nature of these descriptions has two benefits: firstly, workflows can be ported to different computing infrastructures and secondly, these templates can be shared between users working on the same field of interest. Listing 1 shows a JSON template of a test workflow. According to this template, the workflow executes one stage named *process0* that must be executed in the front-end machine (*ramses*) using as Operating System the *64-bit* version of *Ubuntu*. The requirements of the process are deploying *4 single-core* machines with *4GB* of memory and one disk of *20 GB*. The execution of the stage requires invoking, for each node, the program *test* using as argument the filename of *one* file contained in *input0*. The output of the stage (and also of the workflow) are all .txt files generated by the program. Notice that all the values that start with # are references to JSON objects defined in the same file or the resource configuration file described below.

```
{
  "stages": [
    {
      "id": "process0",
      "hostId": "#ramses",
      "environmentId": "#ubuntu64bit",
      "nodes": [
        {
          "numNodes": "4",
          "coresPerNode": "1",
          "memorySize": "4096m",
          "disks": [
            {
              "nDisk": "0",
              "diskSize": "20g"
            }
          ]
        }
      ],
      "execution": [
        {
          "path": "./test",
          "arguments": "#input0(1)"
        }
      ],
      "stageIn": [
        {
          "id": "#input0"
        }
      ],
      "stageOut": [
        {
          "id": "output0",
          "type": "File",
          "filterIn": "*.txt",
          "replica": "none"
        }
      ]
    }
  ]
}
```

Listing 1: Workflow template example

### 3.3.2 Resource information file

To convert the abstract workflow into a concrete workflow or executable workflow, the WMS needs information about the hosts, the environments and the input files. This information can be found in a configuration file like the one showed in Listing 2 for the previous workflow. The array *hosts* contains a list with the data for connecting to the front-end hosts such as: the host name, port and different credentials depending on the platform to be used (for instance, a certificate for Windows Azure and a user/password pair for a cluster). *Environments* is an ad-hoc field for cloud platforms that defines the required features of the VMI (Virtual Machine Image) to use as a base to create the VMs and the software packages that should be installed on it. VMIs are obtained from the image repository associated to each deployment. Finally, *inputFiles* declares the input files of the workflow: the identifier, the type (File, Parameter, etc.) and the physical location (URI).

```
{
  "hosts": [
    {
      "hostId": "ramses",
      "type": "Cloud",
      "subType": "OpenNebula",
      "hostName": "ramses.i3m.upv.es",
      "port": "1111",
      "credentials": {
        "userName": "userName",
        "password": "passWord"
      }
    }
  ],
  "environments": [
    {
      "environmentId": "ubuntu64bit",
      "osName": "linux",
      "arch": "x86_64",
      "osFlavour": "ubuntu",
      "osVersion": "14.04",
      "packages": [
        "unzip"
      ]
    }
  ],
  "inputFiles": [
    {
      "id": "input0",
      "type": "File",
      "values": [
        "db.zip"
      ],
      "extract": "true"
    }
  ]
}
```

Listing 2: Configuration file

### 3.4 Workflow validation

The first step is parsing and validating the workflow template and the resource information file with a JSON processor. Once both documents have been analysed, the next step is to carry out the semantic validation by cross-validating the information provided in both files. The WMS implements a semantic validator which checks if the documents meet different rules. For instance, some fields only allow concrete types of values (memory is an integer followed by the characters 'm', 'g' or 't'). Moreover, every reference in the workflow template should exist in the resource configuration file. If any rule is violated, the system prompts to the user the erroneous file and line.

### 3.5 Workflow planning

The mapping or planning process distinguishes our WMS from other systems by providing a novel approach that dynamically provisions cloud computing resources. It uses the information of the resource configuration file for transforming the abstract workflow into an executable workflow. The mapper component of the system adds tasks for deploying cloud resources only when they are needed and tasks for undeploying them when their outputs have been staged-out. In addition, data management tasks are added for data staging in/out the required input by the tasks or output to the user selected location, respectively. Figure 2 shows the mapping process of a simple workflow with three tasks (A, B and C) to an executable workflow where A and C are deployed on cloud resources and B in a cluster. Although next subsection gives an extensive explanation of each task, the planner produces four types of nodes: deploy, copy, undeploy, cleanup and copyout.

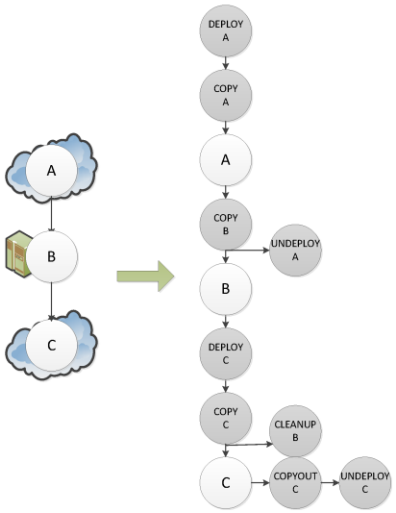


Fig. 2: Planner conversions.

### 3.6 Workflow execution

Once the mapper has produced the executable, it is submitted to the workflow execution engine. The execution of the workflow begins with the initialization of every element: the state of the tasks are set to *IDLE* and the state of the inputs/outputs to *DISABLED*. Next, due to the data-flow nature of the workflow system, the inputs provided by the user are *ENABLED*, allowing the execution of the first task(s). The workflow execution engine is controlled by two core functions: *runTask* and *getStatus*. The runtime checks if all the inputs of a task are enabled, calling *runTask* in that case. When a task is submitted, the engine periodically monitors its status through the *getStatus* function and if it has finished successfully, enables the outputs of the tasks (which in turn are normally inputs of the next tasks). Obviously, the behaviour of runTask and getStatus will vary according to infrastructure (cluster and cloud) and the task type (deploy, copy, user-defined, undeploy, cleanup or copyout).

#### 3.6.1 Deploy task execution

The execution of a deploy task is required when the user desires to execute a task of the abstract workflow in a cloud platform. In order to dynamically deploy cloud

computing resources, the system makes a request to the IM (Infrastructure Manager) [9]. The main function of the IM is to deploy and automatically configure the virtual infrastructure required to execute and manage an application in a cloud computing environment, expressed in a RADL (Resource and Application Description Language) document.
On the one hand, the *runTask* function in a deploy task calls the deploy function of the IM API. Prior to it, the system needs to build a RADL document, using the hardware and software requirements of the task expressed in the JSON document. By means of a RADL document, the WMS calls the IM to configure the deployment as a Portable Batch System (PBS) cluster where all nodes share the same disk via NFS. In this manner, PBS acts as the scheduler of the jobs that the task should execute. On the other hand, the *getStatus* invokes the API function that queries the status of the infrastructure. The task is considered to be finished when the status returned by the API is *configured*. From this point on, the WMS interacts with the cloud infrastructure through SSH, using the information returned by the API (public IP and user credentials).

#### 3.6.2 Copy task execution

The copy task is in charge of the data management during the execution, one of the most crucial parts of any WMS. Moreover, these tasks are executed regardless of the computing platform used (cluster or cloud). With respect to the input data, the system can download any file that can be retrieved with the protocols supported by the unix *wget* command (http, https and ftp). Another important feature is the possibility of explicitly indicating that the input files should be extracted on the destination resources. However, since there are tools that require compressed data as input, this extraction should be optional. In any case, the stage-in of an input file triggers the submission of a job to the physical or virtual cluster scheduler for downloading the file and next, if it is required, extracting the file.
The other type of stage-ins are the intermediate results produced by previous tasks in the DAG. To handle the transference of this kind of data, the WMS submits a basic job that invokes the scp (Secure Copy Protocol) program with the corresponding credentials and arguments.
The goal of *getStatus* in a copy task is to make sure that all the jobs submitted by *runTask* have finished successfully.

#### 3.6.3 User-defined task execution

In contrast to the previous tasks, user-defined tasks are the same that appear in the abstract workflow specification but now they are executable. In our WMS, a user-defined task is said to be executable when two conditions are met: firstly, the target infrastructure is already available (the cluster is accessible or the cloud computing platform is deployed), and secondly, the input data needed by the tasks has been staged-in to these resources. As it can be appreciated, both conditions correspond to the actions performed by the deploy task and copy task, respectively.
According to the abstract workflow, a task can contain a block of executions or commands to execute. When the *runTask* function is invoked for this kind of tasks, the WMS analyses the commands to determine if there is parallelism in the submission of the job or not. The parallelism of a task is explicitly indicated by the user in the abstract workflow, appending the "*(x)*" expression to an argument where $x$ is the granularity. The granularity refers to the number of files processed per computing node.

#### 3.6.4 Undeploy task execution

As in the deployment task execution case, the *runTask* function calls the proper function of the IM API, *destroyInfrastructure*.
The aim of *getStatus* in this case is to make sure that the infrastructure removal operation is correctly carried out. This is especially important when public clouds are used to avoid incurring in unnecessary costs.

#### 3.6.5 Cleanup task execution

The cleanup task is the equivalent of the undeploy task but for the case of clusters. Because a workflow task usually generates large of amounts of data and clusters are infrastructures shared with other users, a best practice consists on cleaning up the data once it has been staged-out. Thus, the function *runTask* simply deletes via SSH the whole execution directory created for the task and *getStatus* makes sure that the operation is actually done.

#### 3.6.6 Copyout task execution

From the user's point of view, the purpose of the copyout tasks is to retrieve the data products of the computations. The mapper attaches these special tasks only to the final tasks of the abstract workflow specification (i.e tasks which don't have dependencies with other tasks).
The *runTask* function starts the stage-out of the output to one or more locations. The default action is to transfer the data to the user local space (where the submit host is being executed). If besides the field *replica* of the output contains references to another data storage sites, the data will be also copied to these locations. The other function, *getStatus*, will monitor the data transference until all of them are completed.

### 3.7 Fault tolerance

Although workflow execution failures in clusters and Cloud Computing platforms are not very common, fault tolerance is of most importance in distributed computing environments. It is necessary to provide the proper fault-tolerance mechanisms to handle failures and support the

reliable execution in the presence of software or hardware failures. Due to the difference in terms of requirements between the tasks that compose a workflow, the WMS defines different fault tolerance policies for each task. The policies simply define the number of retries in case of software failure or hardware failure. The user also has the possibility of indicating such values in the abstract workflow specification, using the object *retries* and its fields *OnWallTimeExceeded*, *OnSoftwareFailure* and *OnHardwareFailure* inside a stage object. If, for some reason, a task exceeds the maximum number of retries for any type of failure, the execution of the workflow is automatically aborted.

### 3.8 Provenance

Workflow provenance is critical to users to be able to follow the evolution of their executions and to determine the cause behind a failure. In that sense, the WMS implements a logging system that registers in a file the events that occur during the execution along with other important information (for example, the elapsed time of a task that has finished). This information is not only useful for monitoring the progress of the experiment but also for getting performance statistics as we have done in the use case presented in the next section.

## 4. Use case: Orthosearch

OrthoSearch (Orthologous Gene Searcher) [10] [11] is a genomics comparative workflow. Initially conceived as a Perl-based routine, it is a profile-protein, reciprocal best hits (RBH) based solution for homology inference among species. It comprises several stages and uses distinct bioinformatics tools, such as Mafft [12] and HMMER [13] which confront an orthologous database with an organism multifasta protein data. The workflow structure is depicted in Figure 3.

OrthoSearch was initially built in order to infer homology between Protozoa species, although there is no restriction to any specific genome. It has already been evaluated and proven to be effective when inferring orthology among five Protozoa organisms with two distinct orthologous databases [14], NCBI COG and KOG.

## 5. Experimentation

### 5.1 Data selection

We selected a subset of EggNOG database version 4 [15] which comprises eukaryotic ortholog groups only, EggNOG KOG. Its design comprises up-to-date techniques for ortholog groups construction, inparalogs recognition, robust automatic annotation, single-copy ortholog groups nesting and reconstruction of phylogenetic trees based on the generated clusters multiple alignments.

Three Protozoa species were selected to be confronted with EggNOG KOG database: Cryptosporidium hominis, Entamoeba histolytica and Leishmania infantum.

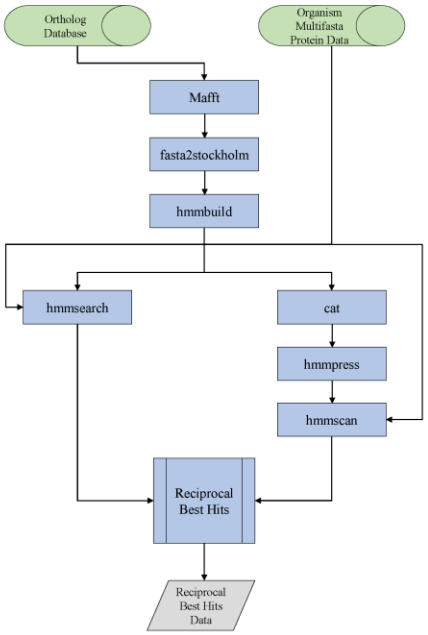


Fig. 3: Orthosearch abstract workflow.

### 5.2 Sequential execution

Figure 3 displays that the Orthosearch pipeline is composed of 8 stages or processes: Mafft, fasta2stockholm, hmmbuild, hmmsearch, cat, hmmpress, hmmscan and Reciprocal_Best_Hits. Initially, the serialized version of this pipeline was executed using a single Ubuntu 14.04 compute instance with 16 CPU cores, 24GB RAM and 100GB disk.

### 5.3 Workflow engine execution

After the sequential execution, the workflow was executed using the WMS presented in this work. The first action that the tool carries out is restructuring the workflow according to the parallelism expressed for each stage on the abstract specification. As a result, the original 8 stages of the pipeline were reduced to 4 stages: Mafft/fasta2stockholm/hmmbuild, hmmsearch, cat/hmmpress/hmmscan and Reciprocal_Best_Hits. On the one hand, Mafft/fasta2stockholm/hmmbuild and hmmsearch are stages that admit trivial parallelism and it fits very well the cloud computing execution model. The resources allocated for this task were 16 Ubuntu 14.04 contextualized VMs on a public cloud (supported by OpenNebula). On the other hand, cat/hmmpress/hmmscan requires the execution of a specific parallel version of hmmscan, implemented on MPI. For that reason, a cluster (kahan.dsic.upv.es) was selected for executing this part of the workflow with 6 concurrent processes. The last stage, Reciprocal_Best_Hits

cannot be parallelized and so was executed on a VM with hardware features similar to the one used on the sequential execution.

### 5.4 Performance results

Table 1 shows a comparison between the response time on the sequential case and the WMS execution for each protozoic organism used.

Table 1: Response time for each scenario.

| | Cryptosporidium | Entamoeba | Leishmania |
|---|---|---|---|
| Total time | 414 minutes | 611 minutes | 575 minutes |
| Speed-up | 5X | 3,84X | 4,13X |

In our best scenario, the WMS provided a 5.0 speedup ratio, with a total 6 hours and 54 minutes of execution time. The same experiment, in a sequential approach, required 34 hours and 10 minutes.

## 6. Conclusions and future directions

The advent of Cloud Computing and its core characteristics (rapid elasticity, resource pooling, and pay-per-use, among others) are well-suited to the nature of scientific applications that experience a variable demand during execution. As a consequence, many WMSs derived from projects in the area of grid computing were updated to support the execution on Cloud resources. However, many of their features are optimized for grids and thus are unable to obtain the most key aspects of clouds, such as dynamic provisioning of resources. For that reason, this work presents a novel multi-platform (clusters and clouds) WMS with support for on-demand provisioning of multi-cloud (public, private and hybrid) customized cloud computing resources.
The tool developed in this work has been tested using a comparative genomics pipeline, called Orthosearch. Promising results were obtained, with significant speedup ratio compared to the batch-oriented pipeline. The current working lines include adding support for Grid computing, using a more efficient transference protocol than SSH and implementing data privacy.

## Acknowledgments

This paper wants to acknowledge the support of the EUBrazilCC project, funded by the European Commission (STREP 614048) and the Brazilian MCT/CNPq N° 13/2012, for the use of its infrastructure. The authors would like also to thank the Spanish "Ministerio de Economía y Competitividad" for the project "Clusters Virtuales Elásticos y Migrables sobre Infraestructuras Cloud Híbridas" with reference TIN2013-44390-R.

## References

[1] E. Deelman, D. Gannon, M. Shields, and I. Taylor, "Workflows and e-science: An overview of workflow system features and capabilities," *Future Generation Computer Systems*, vol. 25, no. 5, pp. 528–540, 2009.

[2] C. Hoffa, G. Mehta, T. Freeman, E. Deelman, K. Keahey, B. Berriman, and J. Good, "On the Use of Cloud Computing for Scientific Workflows," in *2008 IEEE Fourth International Conference on eScience*, pp. 640–645, IEEE, Dec. 2008.

[3] E. Deelman, K. Vahi, G. Juve, M. Rynge, S. Callaghan, P. J. Maechling, R. Mayani, W. Chen, R. Ferreira da Silva, M. Livny, and K. Wenger, "Pegasus, a workflow management system for science automation," *Future Generation Computer Systems*, vol. 46, pp. 17–35, May 2015.

[4] K. Wolstencroft, R. Haines, D. Fellows, A. Williams, D. Withers, S. Owen, S. Soiland-Reyes, I. Dunlop, A. Nenadic, P. Fisher, J. Bhagat, K. Belhajjame, F. Bacall, A. Hardisty, A. Nieva de la Hidalga, M. P. Balcazar Vargas, S. Sufi, and C. Goble, "The Taverna workflow suite: designing and executing workflows of Web Services on the desktop, web or in the cloud.," *Nucleic acids research*, vol. 41, pp. W557–61, July 2013.

[5] C. A. Goble, J. Bhagat, S. Aleksejevs, D. Cruickshank, D. Michaelides, D. Newman, M. Borkum, S. Bechhofer, M. Roos, P. Li, and D. De Roure, "myExperiment: a repository and social network for the sharing of bioinformatics workflows.," *Nucleic acids research*, vol. 38, pp. W677–82, July 2010.

[6] I. Altintas, C. Berkley, E. Jaeger, M. Jones, B. Ludascher, and S. Mock, "Kepler: an extensible system for design and execution of scientific workflows," in *Proceedings. 16th International Conference on Scientific and Statistical Database Management, 2004.*, pp. 423–424, IEEE, 2004.

[7] J. Goecks, A. Nekrutenko, and J. Taylor, "Galaxy: a comprehensive approach for supporting accessible, reproducible, and transparent computational research in the life sciences.," *Genome biology*, vol. 11, p. R86, 2010.

[8] J. Yu and R. Buyya, "A Taxonomy of Workflow Management Systems for Grid Computing," *Journal of Grid Computing*, vol. 3, pp. 171–200, Jan. 2006.

[9] M. Caballer, I. Blanquer, G. Moltó, and C. de Alfonso, "Dynamic Management of Virtual Infrastructures," *Journal of Grid Computing*, Apr. 2014.

[10] S. M. S. da Cruz, M. Mattoso, V. Batista, A. M. R. Dávila, E. Silva, F. Tosta, C. Vilela, M. L. M. Campos, R. Cuadrat, and D. Tschoeke, "OrthoSearch," in *Proceedings of the 2008 ACM symposium on Applied computing - SAC '08*, (New York, New York, USA), p. 1282, ACM Press, Mar. 2008.

[11] S. M. S. da Cruz, V. Batista, E. Silva, F. Tosta, C. Vilela, R. Cuadrat, D. Tschoeke, A. M. R. Dávila, M. L. M. Campos, and M. Mattoso, "Detecting distant homologies on protozoans metabolic pathways using scientific workflows.," *International journal of data mining and bioinformatics*, vol. 4, pp. 256–80, Jan. 2010.

[12] K. Katoh and D. M. Standley, "MAFFT multiple sequence alignment software version 7: improvements in performance and usability.," *Molecular biology and evolution*, vol. 30, pp. 772–80, Apr. 2013.

[13] R. D. Finn, J. Clements, and S. R. Eddy, "HMMER web server: interactive sequence similarity searching.," *Nucleic acids research*, vol. 39, pp. W29–37, July 2011.

[14] R. R. C. Cuadrat, S. M. da Serra Cruz, D. A. Tschoeke, E. Silva, F. Tosta, H. Jucá, R. Jardim, M. L. M. Campos, M. Mattoso, and A. M. R. Dávila, "An orthology-based analysis of pathogenic protozoa impacting global health: an improved comparative genomics approach with prokaryotes and model eukaryote orthologs.," *Omics : a journal of integrative biology*, vol. 18, pp. 524–38, Aug. 2014.

[15] S. Powell, K. Forslund, D. Szklarczyk, K. Trachana, A. Roth, J. Huerta-Cepas, T. Gabaldón, T. Rattei, C. Creevey, M. Kuhn, L. J. Jensen, C. von Mering, and P. Bork, "eggNOG v4.0: nested orthology inference across 3686 organisms.," *Nucleic acids research*, vol. 42, pp. D231–9, Jan. 2014.

## 8.6 Anexo F – Rotina (*script*) para conversão dos arquivos em formato *multifasta* para formato *stockholm*

```
#!/usr/bin/env perl -w

my $usage = "Usage: $0 <gapped FASTA alignment file(s)>\n";

my @argv;
while (@ARGV) {
    my $arg = shift;
    if ($arg =~ /^-/) {
if ($arg eq "-h") { print $usage; exit }
else { die $usage }
    } else {
push @argv, $arg;
    }
}
push @argv, "-" unless @argv;

# loop through FASTA files
foreach my $fasta (@argv) {
# read FASTA file
    my %seq;
    my @name;
    my $name;
    open FASTA, "<$fasta" or die "Couldn't open '$fasta': $!";
    while (<FASTA>) {
if (/^\s*>\s*(\S+)/) {
$name = $1;
die "Duplicate name: $name" if defined $seq{$name};
push @name, $name;
} else {
if (/\S/ && !defined $name) {
warn "Ignoring: $_";
} else {
s/\s//g;
```

```
$seq{$name} .= $_;
}
}
    }
    close FASTA;

# check all seqs are same length
    my $length;
    my $lname;
    foreach my $name (@name) {
my $l = length $seq{$name};
if (defined $length) {
die "Sequences not all same length ($lname is $length, $name is
$l)" unless $length == $l;
} else {
$length = length $seq{$name};
$lname = $name;
}
    }

# print Stockholm output
    print "# STOCKHOLM 1.0\n";
    foreach my $name (@name) {
print $name, " ", $seq{$name}, "\n";
    }
    print "//\n";
}
```

## 8.7 Anexo G – Aplicação para identificação de melhores *hits* recíprocos no elastic-OrthoSearch

```
Classe RUN

require 'Parser'
require 'csv'
require 'BestHits'
require 'HmmerHit'
require 'Hits'
require 'optparse'



options = {}
OptionParser.new do |opts|
  opts.banner = "Usage: example.rb [options]"

  opts.on("-v n", "--[no-]verbose", "Run verbosely") do |v|
    options[:verbose] = v
  end
  opts.on("-O n", "--og_folder", "OG Folder") do |og|
    options[:og_folder] = og

  end
  opts.on("-H n", "--hmmscan_input_file", "Hmmscan Input File")
do |h|
    options[:hmmscan_input_file] = h
  end
  opts.on("-C n", "--csv_file", "CSV Output File") do |csv|
    options[:csv_file] = csv
  end
end.parse!

og_folder = options[:og_folder]
hmmscan_input_file = options[:hmmscan_input_file]
csv_file = options[:csv_file]
```

```
searchHits = Array.new
scanHits = Array.new

## instancia objeto da classe best hits, o que automaticamente
chamara o parser para identificar os best hits e guardar os
valores no atributo
## hits do objeto criado
scanHits = Parser.new(hmmscan_input_file.to_s).scanfinalarr()

# vamos iniciar a busca dos melhores hits no hmmsearch
 searchHits = Parser.new(og_folder.to_s).searchfinalarr()

bestHits = Hits.new(scanHits,searchHits).besthits()

first_line = true

csv = CSV.open(csv_file.to_s,"wb") do |csv|
  bestHits.each do |hits|
    if (first_line == true)
      csv << hits.varnames()
      csv << hits.to_s()
      first_line = false
    else
      csv << hits.to_s()
    end
  end

end
```

Classe PARSER

```
class Parser

  attr_accessor :hit, :scanfinalarr, :searchfinalarr, :tmparray
  attr_accessor :stop, :id_query
  arr = Array.new
  @scanfinalarr = Array.new
```

```
  @searchfinalarr = Array.new
  @hit = nil
  @hit = Array.new

  def initialize(parameter)

    if File.file?(parameter)
      parser_init_file(parameter)
      return @scanfinalarr
    elsif File.directory?(parameter)
     parser_init_folder(parameter)
     return @searchfinalarr
    end
  end


  def parser_init_file(parameter)

    @tmparray = Array.new
    @scanfinalarr = Array.new
    @hit = Array.new
    @type = nil
    my_hmmer_input_file = File.new(parameter.to_s)
    my_hmmer_input_file.each_line() do |line|
        @tmparray << parse_line(line)
    end

    @tmparray.each do |test|
       if test.class() == HmmScanBestHits
          if test.instance_variable_defined?(:@idquery)
            @scanfinalarr << test
          end
       end
    end
end

  def parser_init_folder(parameter)
```

```
    @tmparray = Array.new
    @searchfinalarr = Array.new
    Dir.foreach(parameter) do |search|
      next if search == '.' or search == '..'
      # instancia objeto da classe best hits, o que
automaticamente chamara o parser para identificar os best hits
e guardar os valores no atributo hits do objeto criado
      @hit = Array.new
      @type = nil
      $stop = 0
      my_hmmer_input_file                                    =
File.new(File.expand_path(parameter)+"/"+search.to_s())
      my_hmmer_input_file.each_line() do |line|
       if ($stop == 0)
          @tmparray << parse_line(line)
        end
      end
    end

    @tmparray.each do |test|
    if test.class() == HmmSearchBestHits
      if test.instance_variable_defined?(:@idquery)
          @searchfinalarr << test

       end
    end
    end

  end

  private



  def parse_line(line)
      # identifica se vamos ler um arquivo de saida do hmmscan
ou hmmsearch ou se o arquivo eh invalido
      if  line =~ /^#\shmmscan/
        @type = "hmmscan"
```

```
      elsif line =~ /^#\shmmsearch/
        @type = "hmmsearch"
      # se a linha nao comeca com comentario ou espaco vazio
tentaremos verificar o valor da variavel @type
      elsif line =~  /\S/ && line !~ /^#/
      if @type.match("hmmscan")
        # devemos processar o arquivo que veio do hmmscan
        @hit = HmmScanBestHits.new(line)
      elsif @type.match("hmmsearch")
        # devemos processar o arquivo que veio do hmmsearch
        @hit = HmmSearchBestHits.new(line) unless ($stop == 1)
      else
        # arquivo eh invalido para processamento
        raise ArgumentError, "Wrong input file. Try again"
      end # end if
      end # end if
      return @hit
   end # end parse line

end # eof class Parser
```

Classe HMMER3HIT

```
class Hmmer3Hit

  def initialize
    # This is an abstract class. Prevents 'new' being called on
this class
    # and force implementation of 'initialize' in inheriting
classes.
    raise NotImplementedError
  end

  attr_reader   :full_sequence_e_value,   :full_sequence_score,
:full_sequence_bias
  attr_reader   :best_1_domain_e_value,   :best_1_domain_score,
:best_1_domain_bias
  attr_reader :best_domain_exp, :best_domain_n
```

```
end # class Hmmer3hit

class HmmScanBestHits < Hmmer3Hit
    # Sets hit data.

  # class atributes
     attr_reader :best_model, :idquery
     # eof class atributes

    def initialize(line) # class constructor
      seek_line_hit(line) # calls for this method to seek the
best hit
    end #initialize

    def seek_line_hit(line)
      #  captures  gi  and  accession  information.  sample...
below...
      # Query:       gi|209882739|ref|XP_002142805.1|  [L=663]
      if  line =~ /^Query.\s+(\S+).+$/
              @@id_query = $1
            else
              # seeks for each enumerated value as shown below.
stores them at several private attributes
              # --- full sequence ---    --- best 1 domain ---
-#dom-
              #E-value   score   bias     E-value   score   bias
exp  N  Model         Description
              #    ------- ------ -----     ------- ------ -----
---- --  --------      -----------
              #    1e-140   460.3   0.0   1.2e-140   460.2   0.0
1.0  1  COG0018.mafft
              if             line             =~/^\s+(\d+\S+|[0-
9])\s+(\d+.\d+)\s+(\d+.\d+)\s+(\d+\S+|[0-
9])\s+(\d+.\d+)\s+(\d+.\d+)\s+(\d+.\d+)\s+(\d+)\s+(\S+)/
                @idquery = @@id_query
                @full_sequence_e_value = $1.to_f()
                @full_sequence_score = $2.to_f()
```

```
                @full_sequence_bias = $3.to_f()
                @best_1_domain_e_value = $4.to_f()
                @best_1_domain_score = $5.to_f()
                @best_1_domain_bias = $6.to_f()
                @best_domain_exp =  $7.to_f()
                @best_domain_n = $8
                @best_model = $9
               end
      end
    end

  attr_accessor :idquery

  def to_s


           return                  @idquery,         @best_model,
@full_sequence_e_value, @full_sequence_score,
                   @full_sequence_bias,  @best_1_domain_e_value,
@best_1_domain_score,
                   @best_1_domain_bias,        @best_domain_exp,
@best_domain_n
         end

end #eof class HmmScanBestHits




class HmmSearchBestHits < Hmmer3Hit
    # Sets hit data.

     # class atributes
    attr_accessor :best_sequence, :best_description, :idquery
    # eof class atributes

    def initialize(line) # class constructor
       if ($stop == 0)
          seek_line_hit(line)  # calls for this method to seek
```

```
the best hit
       end
    end # eof initialize

    def seek_line_hit(line)
      # captures gi and accession information
      # sample... below...
      # CQuery:       COG0001.mafft  [M=431]
      if  line =~ /^Query.\s+(\S+).+$/
         @@id_query = $1
      else
           # seeks for each enumerated value as shown below.
stores them at several private attributes
           # --- full sequence ---   --- best 1 domain ---    -
#dom-
           # E-value   score   bias      E-value   score   bias
exp  N  Sequence                          Description
           # ------- ------ -----    ------- ------ -----   ---
- --  --------                            -----------
           #  7.4e-38   128.0    0.0      1.1e-37   127.5    0.0
1.1   1   gi|209877426|ref|XP_002140155.1| adenosylmethionine-8-
amino-7-oxo
              if             line             =~/^\s+(\d+\S+|[0-
9])\s+(\d+.\d+)\s+(\d+.\d+)\s+(\d+\S+|[0-
9])\s+(\d+.\d+)\s+(\d+.\d+)\s+(\d+.\d+)\s+(\d+)\s+(\S+)(\S|.+)$
/
                @idquery = @@id_query
                @full_sequence_e_value = $1.to_f()
                @full_sequence_score = $2.to_f()
                @full_sequence_bias = $3.to_f()
                @best_1_domain_e_value = $4.to_f()
                @best_1_domain_score = $5.to_f()
                @best_1_domain_bias = $6.to_f()
                @best_domain_exp =  $7.to_f()
                @best_domain_n = $8
                @best_sequence = $9
                @best_description = $10
                #no  hmmsearch  podemos  ter  varias  linhas  de
```

```
COG's mas somente a primeira interessa, eh o melhor hit do
hmmsearch
                # por isso vamos parar a execucao do loop
                $stop = 1
              end
      end
     end

   def to_s
      return          @idquery,          @full_sequence_e_value,
@full_sequence_score, @full_sequence_bias,
             @best_1_domain_e_value,       @best_1_domain_score,
@best_1_domain_bias,
             @best_domain_exp,  @best_domain_n,  @best_sequence,
@best_description
    end

 end # eof class HmmerSearchBestHits

Classe HITS

class Hits

  attr_accessor :besthits


  def initialize (scanHits,searchHits)


    bestHitsHash = Hash.new
     bestHitsCompleteHash = Hash.new
     bestHitsArray = Array.new

      # vamos agora criar uma hash para guardar os pares
idquery e COG encontrados, que serao utilizados posteriormente
para match com
      # as entradas do hmmsearch
      scanarrQry = Array.new
```

```
      scanHash = Hash.new
      scanCompleteHash = Hash.new

    findScanBestHits(scanHits,    scanHash,    scanCompleteHash,
scanarrQry)

    searcharrQry = Array.new
    searchHash = Hash.new
    searchCompleteHash = Hash.new

 findSearchBestHits(searchHits, searchHash, searchCompleteHash,
searcharrQry)

 findBestHits(scanHash,      scanCompleteHash,      searchHash,
searchCompleteHash, bestHitsHash, bestHitsCompleteHash)

 end

  def  findScanBestHits(scanHits,  scanHash,  scanCompleteHash,
scanarrQry)

           # para cada elemento do array de objetos Parser,
quero pegar apenas os que possuem um idquery efetivo, ou seja,
entradas do arquivo
           # hmmscan que tiveram hit
           scanHits.each do |hit|
             scanarrQry       <<       hit.idquery       unless
hit.idquery.nil?()
           end

           # ordenamos o array e pegamos as entradas unicas
           scanarrQry.sort!
           scanarrQry.uniq!

           # vamos filtrar o array para inserir os dados com
base na idquery
           scanarrQry.each do |query|
              t1 = Thread.new() do #print "."
```

```
                filtered = filterArray(query, scanHits)
                # dummy values
                eval = 100.0
                score = 0
                # eof dummy values
                # tratar os dados do array filtrado... para
cada um dos elementos, verifica...
                filtered.each do |hit|
                   # se o eval for maior que o evalue do
objeto, altera o valor de eval, score e preenche a hash com o
par query e best_model
                   if eval > hit.full_sequence_e_value
                      eval = hit.full_sequence_e_value
                      score = hit.full_sequence_score
                      scanHash[query] = hit.best_model
                      scanCompleteHash[query] = hit
                   else
                      # se o eval for igual, testa o valor do
score
                      if eval == hit.full_sequence_e_value
                         # se o score eh menor, assume o novo
evalue e score para o par da hash
                         if score <= hit.full_sequence_score
                            eval = hit.full_sequence_e_value
                            score = hit.full_sequence_score
                            scanHash[query] = hit.best_model
                            scanCompleteHash[query] = hit
                         end
                      end
                   end
                   t1.join
end
                end
             end
  end
   def        findSearchBestHits(searchHits,        searchHash,
searchCompleteHash, searcharrQry)
```

```
      searchHits.each do |cog|
        t1 = Thread.new() do # print "."
           searchHash[cog.idquery] = cog.best_sequence
           searchCompleteHash[cog.idquery] = cog
            t1.join
         end
      end
      end


   def   findBestHits(scanHash,   scanCompleteHash,   searchHash,
searchCompleteHash, bestHitsHash, bestHitsCompleteHash)

     @besthits = Array.new

     scanHash.each do |query, cog|
       t1 = Thread.new() do  #print "."
       if query == searchHash[cog]
        bestHitsHash[cog] = query
        @besthits                                                  <<
BestHits.new(scanCompleteHash,searchCompleteHash, cog, query)
      end
      t1.join
     end
   end
    return @besthits
 end

  def filterArray(value, hitArray)

    result = Array.new()
    hitArray.each do |filter|
      result << filter if filter.idquery == value
    end
    return result
  end
end
```

```
Classe BESTHITS

#require 'HmmerHit'
#require 'Hits'
#require 'Parser'

class BestHits



       # atributos que tem origem no hmmscan
        attr_accessor                                :scan_idquery,
:scan_full_sequence_e_value,          :scan_full_sequence_score,
:scan_full_sequence_bias
        attr_accessor               :scan_best_1_domain_e_value,
:scan_best_1_domain_score, :scan_best_1_domain_bias
        attr_accessor                      :scan_best_domain_exp,
:scan_best_domain_n, :scan_best_model
       # eof atributos que tem origem no hmmscan

       # atributos que tem origem no hmmsearch
        attr_accessor                              :search_idquery,
:search_full_sequence_e_value, :search_full_sequence_score
        attr_accessor                :search_full_sequence_bias,
:search_best_1_domain_e_value, :search_best_1_domain_score
        attr_accessor                :search_best_1_domain_bias,
:search_best_domain_exp, :search_best_domain_n
        attr_accessor                      :search_best_sequence,
:search_best_description
       # eof atributos que tem origem no hmmsearch



        def  initialize  (scanCompleteHash,  searchCompleteHash,
cog, query)

          @idquery = cog
          @best_model = query

          @scan_full_sequence_e_value                          =
```

```
scanCompleteHash[query].full_sequence_e_value
          @scan_full_sequence_score                              =
scanCompleteHash[query].full_sequence_score
          @scan_full_sequence_bias                               =
scanCompleteHash[query].full_sequence_bias
          @scan_best_1_domain_e_value                            =
scanCompleteHash[query].best_1_domain_e_value
          @scan_best_1_domain_score                              =
scanCompleteHash[query].best_1_domain_score
          @scan_best_1_domain_bias                               =
scanCompleteHash[query].best_1_domain_bias
          @scan_best_domain_exp                                  =
scanCompleteHash[query].best_domain_exp
          @scan_best_domain_n                                    =
scanCompleteHash[query].best_domain_n

          @search_full_sequence_e_value                          =
searchCompleteHash[cog].full_sequence_e_value
          @search_full_sequence_score                            =
searchCompleteHash[cog].full_sequence_score
          @search_full_sequence_bias                             =
searchCompleteHash[cog].full_sequence_bias
          @search_best_1_domain_e_value                          =
searchCompleteHash[cog].best_1_domain_e_value
          @search_best_1_domain_score                            =
searchCompleteHash[cog].best_1_domain_score
          @search_best_1_domain_bias                             =
searchCompleteHash[cog].best_1_domain_bias
          @search_best_domain_exp                                =
searchCompleteHash[cog].best_domain_exp
          @search_best_domain_n                                  =
searchCompleteHash[cog].best_domain_n
          @search_best_description                               =
searchCompleteHash[cog].best_description


        end
```

```
        def to_s

          return                  @idquery,         @best_model,
@scan_full_sequence_e_value, @scan_full_sequence_score,
                  @scan_full_sequence_bias,
@scan_best_1_domain_e_value, @scan_best_1_domain_score,
                  @scan_best_1_domain_bias,
@scan_best_domain_exp, @scan_best_domain_n,
                  @search_full_sequence_e_value,
@search_full_sequence_score, @search_full_sequence_bias,
                  @search_best_1_domain_e_value,
@search_best_1_domain_score, @search_best_1_domain_bias,
                  @search_best_domain_exp,
@search_best_domain_n, @search_best_description

        end
        def varnames

          return              "BEST_MODEL",              "QUERY",
"HMMSCAN_FULL_SEQUENCE_E_VALUE", "HMMSCAN_FULL_SEQUENCE_SCORE",
          "HMMSCAN_FULL_SEQUENCE_BIAS",
"HMMSCAN_BEST_1_DOMAIN_E_VALUE", "HMMSCAN_BEST_1_DOMAIN_SCORE",
          "HMMSCAN_BEST_1_DOMAIN_BIAS",
"HMMSCAN_BEST_DOMAIN_EXP", "HMMSCAN_BEST_DOMAIN_N",
          "HMMSEARCH_FULL_SEQUENCE_E_VALUE",
"HMMSEARCH_FULL_SEQUENCE_SCORE",
"HMMSEARCH_FULL_SEQUENCE_BIAS",
          "HMMSEARCH_BEST_1_DOMAIN_E_VALUE",
"HMMSEARCH_BEST_1_DOMAIN_SCORE",
"HMMSEARCH_BEST_1_DOMAIN_BIAS",
          "HMMSEARCH_BEST_DOMAIN_EXP",
"HMMSEARCH_BEST_DOMAIN_N", "HMMSEARCH_BEST_DESCRIPTION"
        end
end
```

## 8.8 Anexo H - Exemplo de arquivo CSV, resultado da execução do elastic-Orthosearch

| |
|---|
| BEST_MODEL,QUERY,HMMSCAN_FULL_SEQUENCE_E_VALUE,<br>HMMSCAN_FULL_SEQUENCE_SCORE,HMMSCAN_FULL_SEQUENCE_BIAS,<br>HMMSCAN_BEST_1_DOMAIN_E_VALUE,HMMSCAN_BEST_1_DOMAIN_SCORE,<br>HMMSCAN_BEST_1_DOMAIN_BIAS,HMMSCAN_BEST_DOMAIN_EXP,<br>HMMSCAN_BEST_DOMAIN_N,<br>HMMSEARCH_FULL_SEQUENCE_E_VALUE,<br>HMMSEARCH_FULL_SEQUENCE_SCORE,<br>HMMSEARCH_FULL_SEQUENCE_BIAS,<br>HMMSEARCH_BEST_1_DOMAIN_E_VALUE,HMMSEARCH_BEST_1_DOMAIN_SCORE,<br>HMMSEARCH_BEST_1_DOMAIN_BIAS,HMMSEARCH_BEST_DOMAIN_EXP,<br>HMMSEARCH_BEST_DOMAIN_N,HMMSEARCH_BEST_DESCRIPTION |
| ORTHOMCL18921,gi\|67624699\|ref\|XP_668632.1\|,2.5e-156,518.0,11.1,2.8e-156,517.9,11.1,1.0,1,3.6e-157,518.0,11.1,4.0e-157,517.9,11.1,1.0,1,<br>hypothetical protein<br>KOG3872,gi\|67585174\|ref\|XP_665091.1\|,6.7e-22,78.7,11.7,8.3e-22,78.4,11.7,1.1,1,9.7e-23,78.7,11.7,1.2e-22,78.4,11.7,1.1,1,<br>hypothetical protein<br>K10592.cdhit,gi\|67600586\|ref\|XP_666349.1\|,1.5e-247,825.7,13.7,1.7e-247,825.5,13.7,1.0,1,2.1e-248,825.7,13.7,2.5e-248,825.5,13.7,1.0,1,<br>ubiquitin-protein ligase 1<br>ORTHOMCL3009,gi\|67624323\|ref\|XP_668444.1\|,9.5e-79,264.7,22.1,1.1e-78,264.6,22.1,1.0,1,1.4e-79,264.7,22.1,1.5e-79,264.6,22.1,1.0,1, sf-assemblin-like protein |

## 8.9 Anexo I - Exemplo de arquivo JSON utilizado pelo elastic-OrthoSearch para a especificação de parâmetros de configuração da infraestrutura, estágios de execução e parâmetro de elasticidade

```
{
"hosts":[
	{
		"hostId": "one1",
		"type": "Cloud",
		"subType": "OpenNebula",
		"hostName": "cloud_front_end",
		"port": "2633",
		"credentials":{
			"userName": "user",
			"password": "password"
		}
	}

],
"environments": [
	{
		"environmentId": "ubuntu64bit",
		"osName": "linux",
		"arch": "x86_64",
		"osFlavour": "ubuntu",
		"osVersion": "14.04",
		"packages":[
			"MAFFT",
			"HMMER",
			"unzip",
			"dos2unix",
			"ruby"
		]
	}
],
"stages":[
	{
		"id": "mafft-fasta2stockholm-hmmbuild",
		"hostId": "#one1",
```

```
"environmentId": "#ubuntu64bit",

"nodes": [
      {
                  "numNodes":"4",
                  "coresPerNode":"1",
                  "memorySize":"2048m",
                  "disks": [
                  {
                        "nDisk": "0",
                        "diskSize":"40g"
                  }
                  ]
      }
],

"execution": [
      {
            "path": "./mafft_st_hmmbuild",
            "arguments": "#input2(1)"
      }
],

"retries" : {
      "onWallTimeExceeded": "0",
      "onSoftwareFailure": "0",
      "onHardwareFailure": "0"
},

"stageIn": [
      {
            "id": "input0",
                  "type": "File",
            "values": [
                  "mafft_st_hmmbuild"
            ]
      },
      {
            "id": "input1",
                  "type": "File",
            "values": [
                  "fasta2stockholm.pl"
            ]
```

```
                },
                {
                        "id": "input2",
                                "type": "File",
                        "values": [
                                "eggnog_kog.zip(extract)"
                        ]
                }
        ],

        "stageOut": [
                {
                        "id": "output0",
                        "type": "File",
                        "filterIn": "*.mafft.st.hmm",
                        "replica": "None"
                }
        ]
    },
    {
        "id": "hmmsearch",
        "hostId": "#one1",
        "environmentId": "#ubuntu64bit",

            "nodes": [
              {
                        "numNodes":"4",
                        "coresPerNode":"1",
                        "memorySize":"4096m",
                        "disks": [
                        {
                                "nDisk": "0",
                                "diskSize":"40g"
                        }
                        ]
                }
        ],

        "execution": [
                {
                        "path": "hmmsearch",
                        "arguments":   "-E   0.1   #output0(1)   #input3   >
#output0(1).#input3"
```

```
                }
            ],

            "retries" : {
                "onWallTimeExceeded": "0",
                "onSoftwareFailure": "0",
                "onHardwareFailure": "0"
            },

            "stageIn": [
                {
                    "id": "#output0"
                },
                {
                    "id": "input3",
                        "type": "File",
                    "values": [
                        "Cryptosporidium-hominis-TU502-cdhit-
s1.fasta"
                    ]
                }
            ],

            "stageOut": [
                {
                    "id": "output1",
                    "type": "File",
                    "filterIn": "*.mafft.st.hmm.*",
                    "replica": "None"
                }
            ]
        },
        {
            "id": "cat",
            "hostId": "#one1",
            "environmentId": "#ubuntu64bit",

              "nodes": [
                {
                        "numNodes":"1",
                        "coresPerNode":"1",
                        "memorySize":"4096m",
                        "disks": [
```

```
                        {
                              "nDisk": "0",
                              "diskSize":"40g"
                        }
                        ]
                  }
            ],

            "execution": [
                  {
                        "path": "cd",
                        "arguments": "output0"
                  },
                  {
                  "path": "ls",
                  "arguments": ". | xargs -n 32 cat > $JOBID/#output2"
                  }
            ],

            "retries" : {
                  "onWallTimeExceeded": "0",
                  "onSoftwareFailure": "0",
                  "onHardwareFailure": "0"
            },

            "stageIn": [
                  {
                        "id": "#output0"
                  }
            ],

            "stageOut": [
                  {
                        "id": "output2",
                        "type": "File",
                        "replica": "None",
                        "values": [
                              "hmmcat.hmm"
                        ]
                  }
            ]
      },
      {
```

```
"id": "hmmerpress-hmmerscan",
"hostId": "#one1",
"environmentId": "#ubuntu64bit",

    "nodes": [
      {
                  "numNodes":"1",
                  "coresPerNode":"1",
                  "memorySize":"4096m",
                  "disks": [
                  {
                        "nDisk": "0",
                        "diskSize":"40g"
                  }
                  ]
      }
],

"execution": [
      {
            "path": "hmmpress",
            "arguments": "#output2"
      },
      {
            "path": "hmmscan",
            "arguments": "#output2 #input4 > #input4.out"
      }
],

"retries" : {
      "onWallTimeExceeded": "0",
      "onSoftwareFailure": "0",
      "onHardwareFailure": "0"
},

"stageIn": [
      {
            "id": "#output2"
      },
      {
            "id": "input4",
                  "type": "File",
            "values": [
```

```
                        "Cryptosporidium-hominis-TU502-cdhit-
s1.fasta"
                  ]
            }
      ],

      "stageOut": [
            {
                  "id": "output3",
                  "type": "File",
                  "filterIn": "*.out",
                  "replica": "None"
            }
      ]
   },
   {
      "id": "best-hits",
      "hostId": "#one1",
      "environmentId": "#ubuntu64bit",

          "nodes": [
            {
                        "numNodes":"1",
                        "coresPerNode":"1",
                        "memorySize":"4096m",
                        "disks": [
                        {
                              "nDisk": "0",
                              "diskSize":"40g"
                        }
                        ]
            }
      ],

      "execution": [
            {
            "path": "ruby",
            "arguments":   "Run.rb   -O   output1   -H   #output3   -C
ko_vs_chominis.csv"
            }
      ],

      "retries" : {
```

```
                "onWallTimeExceeded": "0",
                "onSoftwareFailure": "0",
                "onHardwareFailure": "0"
            },

            "stageIn": [
                {
                    "id": "#output1"
                },
                {
                    "id": "#output3"
                },
                {
                    "id": "input5",
                        "type": "File",
                    "values": [
                        "besthits_src.zip(extract)"
                    ]
                }
            ],

            "stageOut": [
                {
                    "id": "output4",
                    "type": "File",
                    "filterIn": "*.csv",
                    "replica": "None"
                }
            ]
        }
]
}
```

## 8.10 Anexo J - Exemplo de registro de atividades (ARQUIVO DE *LOG*) armazenados em instância MongoDB durante a execução do elastic-OrthoSearch

```
10:33:39.176 [main] DEBUG core.Run - JSON parsing with Jackson library START
10:33:39.463 [main] DEBUG core.Run - JSON parsing with Jackson library FINISH
10:33:39.463 [main] DEBUG core.Run - Validation of workflow START
10:33:39.468 [main] DEBUG core.Run - Validation of workflow FINISH
10:33:39.468 [main] DEBUG core.Run - Replacing static arguments of workflow START
10:33:39.469 [main] DEBUG core.Run - Replacing references FINISH
10:33:39.469 [main] DEBUG core.Run - Conversion from logical workflow to physical workflow START
10:33:39.473 [main] DEBUG core.Run - Conversion from logical workflow to physical workflow FINISH
Apr 05, 2015 10:33:39 AM com.google.code.morphia.logging.MorphiaLoggerFactory chooseLoggerFactory
INFO: LoggerImplFactory set to com.google.code.morphia.logging.jdk.JDKLoggerFactory
10:33:39.773 [main] DEBUG core.Run - The ID of the new workflow is: 1428240819773
10:33:39.774 [main] DEBUG core.Run - initialization of workflow START
10:33:39.774 [main] DEBUG core.Run - initialization of workflow FINISH
10:33:40.807 [main] DEBUG core.Run - deploy_mafft-fasta2stockholm-hmmbuild is running
10:48:41.604 [main] DEBUG core.Run - deploy_mafft-fasta2stockholm-hmmbuild has finished
10:48:41.604 [main] DEBUG core.Run - deploy_mafft-fasta2stockholm-hmmbuild Execution time: 00:15:01
10:48:57.249 [main] DEBUG core.Run - copy_mafft-fasta2stockholm-hmmbuild is running
10:53:59.276 [main] DEBUG core.Run - copy_mafft-fasta2stockholm-hmmbuild has finished
10:53:59.276 [main] DEBUG core.Run - copy_mafft-fasta2stockholm-hmmbuild Execution time: 00:05:17
The number of jobs is 2
10:59:12.645 [main] DEBUG core.Run - mafft-fasta2stockholm-hmmbuild is running
11:04:29.220 [main] DEBUG core.Run - deploy_hmmsearch is running
11:04:30.151 [main] DEBUG core.Run - deploy_cat is running
11:09:46.616 [main] DEBUG core.Run - mafft-fasta2stockholm-hmmbuild has finished
11:09:46.616 [main] DEBUG core.Run - mafft-fasta2stockholm-hmmbuild Execution time: 00:10:47
11:19:47.934 [main] DEBUG core.Run - deploy_hmmsearch has finished
11:19:47.934 [main] DEBUG core.Run - deploy_hmmsearch Execution time: 00:15:19
11:19:52.068 [main] DEBUG core.Run - copy_hmmsearch is running
11:19:52.295 [main] DEBUG core.Run - deploy_cat has finished
```

```
11:19:52.295 [main] DEBUG core.Run - deploy_cat Execution time: 00:15:23
11:19:53.265 [main] DEBUG core.Run - copy_cat is running
11:35:47.395 [main] DEBUG core.Run - hmmsearch is running
11:35:47.395 [main] DEBUG core.Run - cat is running
11:35:47.396 [main] DEBUG core.Run - copy_hmmsearch has finished
11:35:47.396 [main] DEBUG core.Run - copy_hmmsearch Execution time: 00:15:59
11:35:47.396 [main] DEBUG core.Run - undeploy_mafft-fasta2stockholm-hmmbuild is
running
11:35:47.396 [main] DEBUG core.Run - copy_cat has finished
11:35:47.397 [main] DEBUG core.Run - copy_cat Execution time: 00:15:55
11:41:11.770 [main] DEBUG core.Run - undeploy_mafft-fasta2stockholm-hmmbuild has
finished
11:41:11.771 [main] DEBUG core.Run - undeploy_mafft-fasta2stockholm-hmmbuild
Execution time: 00:05:24
11:41:12.549 [main] DEBUG core.Run - deploy_hmmerpress-hmmerscan is running
11:46:16.776 [main] DEBUG core.Run - hmmsearch has finished
11:46:16.776 [main] DEBUG core.Run - hmmsearch Execution time: 00:10:29
11:46:18.822 [main] DEBUG core.Run - cat has finished
11:46:18.823 [main] DEBUG core.Run - cat Execution time: 00:10:31
11:56:19.553 [main] DEBUG core.Run - deploy_hmmerpress-hmmerscan has finished
11:56:19.554 [main] DEBUG core.Run - deploy_hmmerpress-hmmerscan Execution time:
00:15:07
11:56:23.662 [main] DEBUG core.Run - copy_hmmerpress-hmmerscan is running
12:06:36.666 [main] DEBUG core.Run - undeploy_cat is running
12:11:36.667 [main] DEBUG core.Run - hmmerpress-hmmerscan is running
12:11:36.667 [main] DEBUG core.Run - copy_hmmerpress-hmmerscan has finished
12:11:36.668 [main] DEBUG core.Run - copy_hmmerpress-hmmerscan Execution time:
00:15:17
12:11:36.910 [main] DEBUG core.Run - undeploy_cat has finished
12:11:36.911 [main] DEBUG core.Run - undeploy_cat Execution time: 00:05:00
12:16:44.920 [main] DEBUG core.Run - deploy_best-hits is running
12:21:59.347 [main] DEBUG core.Run - hmmerpress-hmmerscan has finished
12:21:59.347 [main] DEBUG core.Run - hmmerpress-hmmerscan Execution time: 00:10:22
12:26:59.854 [main] DEBUG core.Run - deploy_best-hits has finished
12:26:59.855 [main] DEBUG core.Run - deploy_best-hits Execution time: 00:10:15
12:27:08.840 [main] DEBUG core.Run - copy_best-hits is running
12:37:53.519 [main] DEBUG core.Run - undeploy_hmmsearch is running
12:37:53.519 [main] DEBUG core.Run - undeploy_hmmerpress-hmmerscan is running
12:42:53.520 [main] DEBUG core.Run - best-hits is running
12:42:55.561 [main] DEBUG core.Run - copy_best-hits has finished
12:42:55.561 [main] DEBUG core.Run - copy_best-hits Execution time: 00:15:55
12:42:55.772 [main] DEBUG core.Run - undeploy_hmmsearch has finished
12:42:55.773 [main] DEBUG core.Run - undeploy_hmmsearch Execution time: 00:05:02
12:42:56.020 [main] DEBUG core.Run - undeploy_hmmerpress-hmmerscan has finished
12:42:56.021 [main] DEBUG core.Run - undeploy_hmmerpress-hmmerscan Execution time:
00:05:02
12:48:07.305 [main] DEBUG core.Run - copyout_best-hits is running
```

```
12:53:09.366 [main] DEBUG core.Run - best-hits has finished
12:53:09.367 [main] DEBUG core.Run - best-hits Execution time: 00:10:15
12:53:11.388 [main] DEBUG core.Run - copyout_best-hits has finished
12:53:11.388 [main] DEBUG core.Run - copyout_best-hits Execution time: 00:05:04
12:53:11.388 [main] DEBUG core.Run - undeploy_best-hits is running
12:58:11.619 [main] DEBUG core.Run - undeploy_best-hits has finished
12:58:11.619 [main] DEBUG core.Run - undeploy_best-hits Execution time: 00:05:00
13:03:11.620 [main] DEBUG core.Run - The execution of core.Run has finished correctly
```

## 8.11 Anexo K - Exemplo de arquivo *multifasta* com proteínas representativas da base de ortólogos EggNOG KOG, utilizado para a criação de nova base de ortólogos

>gi|71905128|ref|XP_830034.1| dynein heavy chain [Trypanosoma brucei TREU927]
MVSAEKDISDVCVDVAGETLKQLGASVSSTNRRSDVVELQIAVGSEKCPPIAAALFSGFVKAACGSEAGTTKVGLTDVNI
DARENFTVLRFAEAVRVRDLDFNELPGDSPPITSVVEVTYMRYLIKHSELQVFQVSDFPKCLTPRGGQRLDTVVAGYVLF
SLQSSEEDDTKVLLYMKSLRRVLGGMCTSSFAETVLSQAWGNMDSVEPSHVTDMLEDAQMAITDFWNDSDQDTAIRTKVV
FLMKTVGSNLREYFAKKTLDAGGVFSGSKNITEVALGCCDEWVNMCKRLTTVDWGSSWGTAFEDVQLITVRDRLSVVVSI
RDLVDEIVELLTAADKLQSLRTETLWETFDSLDIFATTPAVEQLWTNCLDAFYRRLQPVEHRCASALSDFFGERGNLAPQ
TILNEVVKFRQLIRRPAVSKELVNERDALLAKLNERLQSIRMEFERRSESVEDDVALDEEDRRCQAGRFM
>gi|157437596|gb|EDO81805.1| Protein 21.1 [Giardia lamblia ATCC 50803]
MTARCNGITALLISAYKGCHTCLIHLASEMGIQSWSETRALMFAVNSGHEKCARLLLPEMGIQRHDGMTTLMLAAELGNV
NCVKMLIDCEKEMQTKQK
>gi|123485095|ref|XP_001324416.1| hypothetical protein [Trichomonas vaginalis G3]
MTPLIQASYNGNLEVVKYLISVGADKEAKNNDGWTPLIQASFEGHLEIVKYLISVGANKQARDNVGRTSLYVSREEVKNY
LISIGAQ
>gi|123976338|ref|XP_001330496.1| hypothetical protein [Trichomonas vaginalis G3]
MKGVKQDVIALTANEMFPEAIQCYYNEDKNIHIPVGTYDKDSEFSLMKLYKALHDGQEIGFDLCKSSSPCFAEKFNFSIT
TKTSFIRKLKF

## 8.12 Anexo L - Amostra dos grupos ortólogos da base “KO + EggNOG KOG + ProtozoaDB” que obtiveram *hit* contra o proteoma humano

```
K00001.cdhit:

lcl|dme:Dmel_CG3481
lcl|dpo:Dpse_GA17214
lcl|dan:Dana_GF14888
lcl|der:Dere_GG25120
lcl|dpe:Dper_GL25993
lcl|dse:Dsec_GM15656
lcl|dwi:Dwil_GK18290
lcl|dya:Dyak_GE19037
lcl|dgr:Dgri_GH13025
lcl|dmo:Dmoj_GI17643
lcl|dvi:Dvir_GJ18208
lcl|ath:AT1G32780
lcl|ath:AT1G64710
lcl|ath:AT1G77120
lcl|ath:AT5G24760
lcl|pop:POPTR_1075155
…
K00051.cdhit:

lcl|cgl:NCgl0631

K00052.cdhit:

lcl|ath:AT1G31180
lcl|ath:AT1G80560
lcl|ath:AT5G14200
lcl|pop:POPTR_1067569
lcl|rcu:RCOM_0011000
lcl|vvi:100259968
…

KOG0043:

lcl|436017.A4S6M0
lcl|281687.CJA21620
lcl|164328.JGI93960
lcl|135651.CBN08579
lcl|121224.XP_002430528
lcl|99883.ENSTNIP00000015265
lcl|81824.JGI33364
…
KOG4100:

lcl|665079.A7F9H9
lcl|644223.PAS_chr1-1_0350
lcl|573826.CD36_60150
lcl|559307.ZYRO0F17600g
```

```
lcl|559295.KLTH0F01408g
lcl|500485.B6H0Q6
…
ORTHOMCL10002:

gi|146079823|ref|XP_001463872.1|
gi|154333452|ref|XP_001562983.1|
gi|68124838|emb|CAJ02749.1|
gi|71659061|ref|XP_821256.1|
gi|71755631|ref|XP_828730.1|

ORTHOMCL10003:

gi|146079795|ref|XP_001463864.1|
gi|154333436|ref|XP_001562975.1|
gi|68124828|emb|CAJ02694.1|
gi|71665688|ref|XP_819811.1|
```