PUC

ISSN 0103-9741

Monografias em Ciência da Computação
n° 03/11

Processamento de dados Semânticos na Cloud: um estudo de caso com o Protein World Database

Carlos Juliano Moura Viana
Sérgio Lifschitz
Antonio Basílio de Miranda
Edward Hermann Haeusler

Departamento de Informática

PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO DE JANEIRO

RUA MARQUÊS DE SÃO VICENTE, 225 - CEP 22453-900

RIO DE JANEIRO - BRASIL

Monografias em Ciência da Computação, No. 03/11 ISSN: 0103-9741
Editor: Prof. Carlos José Pereira de Lucena Abril, 2011

Processamento de dados Semânticos: um estudo de caso com o Protein World Database

Carlos Juliano Moura Viana, Sérgio Lifschitz, Edward Hermann Haeusler e Antonio Basílio de Miranda

cviana@inf.puc-rio.br, sergio@inf.puc-rio.br, herman@inf.puc-rio.br, antonio@fiocruz.br

Abstract. the Semantic Web has not only brought many opportunities, but also many other challenges into the data management problem. For instance, biological researchers make their genome findings publicly available, but as much data on the web; those findings are unrelated, and difficult to integrate with other data. In this manner, semantic web technologies could offer possibilities such as automatically infer relationships, which in turn could help to cure diseases. However, as described by Hey et al., scientific data is being generated at exponentially growing rates which makes its processing even more resource consuming. In an effort to assist researchers, this work proposes to make semantic data processing in a flexible and scalable way in order to enable inference over available genomic data. This paper presents a hybrid cloud architecture used for processing and sharing large amounts of biological data while exploiting the MapReduce programming model, and semantic web technologies to enable inference over generated semantic genomic data and related data.
Keywords: Protein World DB, Biological Databases, Gene, Genome, Data Integration

Resumo. A Web Semântica não trouxe somente muitas oportunidades, mas também lançou outros desafios na área de gerenciamento de dados. Atualmente, biólogos e demais pesquisadores permitiram que suas descobertas científicas fossem disponíveis publicamente, assim como muitos dados na web. Muitas dessas descobertas não estão relacionadas e dificultam de serem integradas com outras fontes de dados. Desta forma, muitas das tecnologias da web semântica podem oferecer possibilidades ímpares, tais como inferência automática de relacionamentos, os quais poderiam auxiliar na descoberta de cura de doenças. Entretanto segundo Hey[], os dados científicos estão sendo gerados de forma exponencial, assim requerendo ainda mais recursos computacionais para seu processamento e compreensão. Em um esforço para auxiliar os pesquisadores, o presente trabalho propõe realizar o processamento de dados semânticos de uma forma flexível e escalável, e ainda possibilitando fazer inferência sobre os dados genômicos. Este trabalho consiste em uma arquitetura híbrida de Cloud Computing utilizada para o processamento e compartilhamento de grandes volumes de dados biológicos, utilizando o modelo de programação paralelo distribuído MapReduce e também as tecnologias da Web Semântica para permitir a inferência sobre dados genômicos e outros dados ainda não relacionados.
Palavras-chave: Protein World DB, Bancos de dados Biológicos, Gene, Genoma, Integração de Dados

Responsável por publicações:
Rosane Teles Lins Castilho
Assessoria de Biblioteca, Documentação e Informação
PUC-Rio Departamento de Informática
Rua Marquês de São Vicente, 225 - Gávea
22453-900 Rio de Janeiro RJ Brasil
Tel. +55 21 3527-1516 Fax: +55 21 3527-1530
E-mail: bib-di@inf.puc-rio.br
Web site: http://bib-di.inf.puc-rio.br/techreports/

Sumário

Lista de figuras

Lista de tabelas

1 Introdução

A bioinformática é uma área multidisciplinar e convergente que utiliza informações para compreender o domínio da biologia contando com o trabalho de especialistas em métodos estatísticos e de reconhecimento de padrões [1]. Desta forma, a Bioinformática utiliza os conceitos e ferramentas fornecidas pela ciência da computação para análise, manipulação e interpretação dos dados biológicos, tais como, genes, proteínas, estruturas de proteínas, cromossomos, entre outros dados, com a finalidade que os resultados obtidos dessas análises e manipulações possuam relevância biológica [1]. Como em outras áreas do conhecimento, a Bioinformática está sempre buscando novas e rápidas soluções de computação, para de uma forma mais eficiente, seja capaz de interpretar e atribuir significado aos dados.

Em Biologia Molecular muitas atividades de análises estão sendo e muitas delas já foram automatizadas, por exemplo, o procedimento de comparação das sequências entre dois genomas, o sequenciamento de genomas, entre outras. Além disso, o surgimento de novas tecnologias de sequenciamento de genomas tem substancialmente estendido a escala e resolução de muitas aplicações biológicas, assim como tem aumentado a produção de dados da ordem de giga bases (Gbp) por máquina ao dia [2]. Desta forma, essa grande quantidade de dados não somente precisam ser processados, mas também integrados e analisados em uma maneira mais ágil e de uma forma mais confiável.

Os dados biológicos podem ser obtidos a partir de diversos bancos de dados públicos na Web, tais como GenBank, UniProt, Pfam, e em específico o The Protein World Database (ProteinWorldDB) [3]. O ProteinWorldDB é um repositório de dados que contém informações sobre a comparação de genomas do projeto de Comparação de Genomas - GCP [4], entre o Instituto Oswaldo Cruz (FIOCRUZ) e a PUC-Rio. Esse banco contém os resultados da comparação das proteínas de mais do que 400 genomas, contendo dados sobre coordenadas do alinhamento, valor de pontuação (*scores*), dados estatísticos de $E-values$, entre outros [3].

O advento das novas tecnologias de sequenciamento tornou possível a geração de mais dados a uma taxa mais elevada. Por exemplo, os pesquisadores são capazes agora de sequenciar mais de um bilhão de bases de nucleotídeos em um único dia, com um custo relativamente baixo [5]. Esse acontecimento, coloca o sequenciamento em larga escala dentro do alcance de muitos pesquisadores ao redor do mundo. Conseqüentemente, temos um aumento considerável de dados gerados pelas comparações entre essas novas sequencias dos genomas, o que resulta na necessidade de serem processados, e também na necessidade de gerarmos conhecimento (relacionamentos) sobre eles.

Neste trabalho, propomos e descrevemos uma arquitetura para a geração de conhecimento sobre dados biológicos utilizando tecnologias da Web Semântica. Além disso, instanciamos a nossa arquitetura para transformarmos os dados resultantes de comparações de proteínas do banco de dados do Protein World DB em um formato de triplas, com a finalidade de prover uma melhor integração e capacidade de inferência entre esses e outros dados de bancos de dados públicos, tais como Pfam [6], Swiss-Prot [7; 8], KEGG [9], NCBI [10], Taxonomy [11] e Gene Ontology [12].

Este trabalho está organizado da seguinte forma: Na seção 2 descrevemos o problema do gerenciamento de grande volume de dados, o crescimento do volume de dados de sequenciamento, e a utilização do paradigma de *Cloud Computing*. Descrevemos alguns trabalhos relacionados na seção 3. Na seção 4 descrevemos a arquitetura proposta abordando seus componentes. Na seção 5 enumeramos todos os passos envolvidos na implementação da ar-

quitetura proposta, apresentando detalhes sobre a instanciação da arquitetura, resultados, discussão sobre o modelo MapReduce, e também apresentasmos os materiais e métodos. Na seção 6 descrevemos algumas possibilidades de trabalhos que poderiam ser continuados no futuro. Finalmente, na seção 7 apresentamos nossas considerações finais.

2 Descrição do Problema

Nos últimos 5 anos as plataformas de sequenciamento de DNA tem se tornado amplamente disponíveis, reduzindo o custo e o tempo do sequenciamento de DNA em cerca de duas ordens de grandeza, assim possibilitando a democratização desta área de sequenciamento, disponibilizando essa capacidade de sequenciamento, anteriormente somente disponível a grandes centros de pesquisa, nas mãos de diversos pesquisadores espalhados pelo mundo. Imagine esses experimentos científicos gerando bilhões de imagens e outros dados, como por exemplo, de expressão gênica de microarrays espalhados em dezenas de centenas de laboratórios de pesquisa [13].

A área de biologia computacional e bioinformática têm crescido cada vez mais devido principalmente ao aumento da capacidade de geração de dados, através principalmente do aprimoramento das tecnologias de sequenciamento de DNA [13]. A geração de mais dados demandam maiores oportunidades, além de pressionar ainda mais a geração e aprimoramento de métodos de análises e meios de armazenamento desse grande volume de dados. Nenhum outro campo têm maior evidência desta necessidade do que o da "Próxima geração de Sequenciamento" do termo em inglês "Next-generation sequencing", devido principalmente ao aprimoramento das tecnologias de sequenciamento.

As novas tecnologias de sequenciamento, tais como Roche/454, Illumina, SOLiD and Helicos, tem a capacidade de geração de bilhões de "*short reads*", trechos de sequências de DNA com cerca de 50 a 400 pares de bases, na ordem de giga pares de bases para cada máquina ao dia [2]. Com o advento desse volume gigantesco de dados, os pesquisadores vislumbraram desafios referentes a demanda computacional, que ultrapassa a fronteira das infraestruturas locais utilizadas para analisar e armazenar os dados referentes às comparações entre as novas sequências genômicas sequenciadas [14]. Como exemplo do crescimento desta demanda computacional, nos últimos 5 anos o número de genomas têm crescido cerca de 12 vezes, pressionando a criação, reescrita e a elaboração de novos algoritmos para alinhamentos de sequências [2], detecção de proteínas e genes ortólogos [14], além de formas de armazenamento mais eficientes para lidar com esse volume de dados.

Neste momento faz-se importante que os cientistas tenham a compreensão das vantangens e desvantagens de algumas plataformas computacionais de computação de alto desempenho, tais como Grid Computing, Cluster, Cloud Computing e Heterogeneous Computing, a fim de fazer a escolha mais adequada de solução de demanda computaional para o problema em específico. Schadt e colegas [15] trazem uma classificação das abordagens computacionais para lidar com o gerenciamento e análises de dados em larga escala, descrevendo as vantagens e desvantagens de cada abordagem, além de citar exemplos de aplicações adequadas para serem executadas em cada plataforma computacional.

Enquanto a maioria dos pesquisadores de grandes centros de sequenciamento possuem recursos computacionais suficientes para analisar os dados, muitos desses recursos não estão disponíveis para outros pequenos grupos de pesquisadores, o que impede muitas vezes a aplicação das novas tecnologias de sequenciamento. Mesmo entre os grandes centros de pesquisa, o compartilhamento de um grande volume de dados de forma colaborativa, como ocorreu no projeto de sequenciamento de 1000 genomas [16], ainda traz alguns desafios.

O projeto de Comparação de Genomas (GCP) [4] é um outro projeto que trouxe diversos desafios em termos de infraestrutura computacional para a comparação das sequências de mais de 400 organismos selecionados. Como as comparações par-a-par entre as sequências são operações computacionalmente intensivas, foi necessário utilizar uma infraestrutura de

Grid computacional do *World Community Grid* [17] para realizar a tarefa de comparação das sequências na forma todos-contra-todos, entre as proteínas preditas de todos os genomas completamente sequenciados. A matriz de dados resultante gerou uma base de dados da ordem de giga bytes de dados compactados, que poderá se tornar uma base de dados de valor inestimável, a qual poderá ainda ser continuamente incrementada conforme novas sequências genômicas tornarem-se disponíveis, formando o material para diversos estudos pela comunidade científica.

A utilização de uma infraestrutura de Computação na Nuvem (*Cloud Computing*) pode ser uma possível solução para os problemas relacionados a análise e gerenciamento do grande volume de dados gerados pelo GCP, possibilitando a integração desses dados aos dados de outras fontes de dados públicas. A infraestrutura de *Cloud Computing* oferece uma solução conveniente para tratar alguns desafios computacionais trazidos pelas pesquisas atuais em Biologia Molecular, que vão além do que poderiam serem tratados através da plataforma tradicional de *clusters*. Entre os desafios podemos citar os custos associados na operacionalização do cluster, incluindo espaço físico, energia, refrigeração, tolerância a falhas, mecanismos de *backups*, suporte do pessoal de tecnologia da informação, entre outros.

Em *Cloud Computing*, como o método de cobrança é tipicamente limitado ao recurso que utilizamos, isto é, o usuário é cobrado pelos recursos computacionais que têm sido demandados e pelo período de tempo no qual foi utilizado. Além disso, todas as funções administrativas já estão incluídas no custo final. Este modelo *pay-as-you-go* provê uma flexibilidade que é difícil de ser observada nas pesquisas científicas fora dos grandes centros de sequenciamento de genomas ou dos grandes institutos de pesquisa científicas. Ainda disso, este modelo de computação está sendo colocado nas mãos de quaisquer pesquisadores que utilizem os serviços computacionais em nuvem, oferecidos por grandes centros de informação, tais como Google, Amazon ou Microsoft [15].

Além da flexibilidade de utilização e cobrança pelos recursos computacionais utilizados, o modelo de computação em nuvem também possibilita o compartilhamento dos conjuntos de dados e dos resultados de análises entre os pesquisadores. Atualmente muitos conjuntos de dados, tais como, Ensembl e os dados do projeto de 1000 genomas [16], estão sendo disponibilizados através de nuvens públicas. Por exemplo, a Amazon AWS [18] provê um repositório centralizado de conjunto de dados públicos (*Ensembl, 1000 data genome, UniGene* e *Freebase*), onde os dados podem ser compartilhados e integrados em aplicações desenvolvidas para a própria nuvem da Amazon.

Apensar da flexibilidade e da facilidade de acesso a grandes recursos computacionais oferecidos pela computação em nuvem, este modelo ainda não resolve o problema da transferência de dados tanto para dentro e fora da nuvem, quanto entre nuvens. Assim, a tarefa de transferência de grande volumes de dados para o processamento na nuvem pode ser impraticável, trazendo dificuldades para que grupos de pesquisa possam colaborar facilmente entre sí ao utilizarem os serviços da nuvem. Além do problema da transferência de dados, também temos problemas relacionados à privacidade dos dados sendo carregados para o acesso e processamento na nuvem. O trabalho de Schadt e colegas [15] traz outras discussões interessantes sobre a utilização de computação em nuvem e de outras plataformas computacionais para tratar o problema da escolha de uma plataforma mais adequada para o gerenciamento e análise de dados em larga escala.

Este trabalho propõe a descrição de uma arquitetura para possibilitar a integração

semântica dos dados resultantes do projeto de comparação de genomas - GCP [4] com outras fontes de dados públicas, tais como, Pfam, Swiss-Prot, KEGG, RefSeq, entre outras, utilizando a plataforma de computação em nuvem e algumas tecnologias da Web Semântica [19]. Escolhemos esta plataforma computacional devido a considerarmos esta uma melhor opção para permitir a extensão da geração dos resultados de comparação com novas sequências genômicas disponíveis, desde a última comparação [4], além de não termos que nos preocuparmos muito com os problemas de administração da infraestrutura, levando-se em consideração o problemas relacionados ao crescimento exponencial do volume dos dados resultantes das comparações entre as novas sequências dos organismos a serem comparados. Além disso, optamos por esta plataforma computacional devido a possibilidade da utilização das diversas aplicações de bioinformática (*CloudBLAST, CloudBurst, Contrail, Crossbow*, entre outras) que já foram adaptadas e outras que foram desenvolvidas especificamente para lidar com grandes volumes de dados na nuvem [20].

3 Trabalhos Relacionados

A utilização da Web Semântica e de suas ferramentas para integração de fontes de dados não é uma abordagem recente. Newman [21] e colegas descrevem o seu trabalho utilizando o modelo de programação paralelo distribuído *MapReduce* [22] para armazenar dados em um repositório de dados RDF distribuído, com o objetivo de obter escalabilidade semântica utilizando também a computação em nuvem (*Cloud Computing*). Os autores propõem a utilização de uma estrutura denominada de molécula RDF (*RDF Molecule*) para possibilitar a decomposição e junção dos grafos RDF [23] ao utilizar processamento distribuído com auxílio do framework MapReduce, e também para manter a semântica RDF associada. Desta forma, pela decomposição dos grafos RDFs em unidades menores (*RDF molecules*), tais estruturas podem ser indexadas e distribuídas entre os nós do *cluster*. O processamento das consultas é realizado buscando a parte dos dados em cada nó, utilizando a linguagem de consulta SPARQL [24]. O processo de união dos resultados das consultas é realizado obtendo-se os resultados de cada nó e fazendo a junção deles para gerar um resultado final.

Husain e colegas [25; 26; 27] apresentam um *framework* que foi desenvolvido para alavancar o uso do Hadoop [28], uma implementação open-source do *framework* MapReduce, para armazenar e obter um grande número de triplas RDF. Nesse trabalho, descrevem a sua proposta de arquitetura, a forma de armazenamento de dados RDF no sistema de arquivos do Hadoop (HDFS), e também apresentam o algoritmo para responder a consultas SPARQL utilizando *jobs* MapReduce. Essa arquitetura possibilita aos usuários armazenar triplas RDF em *clusters* Hadoop, utilizando *commodities machines*, e ainda responder consultas complexas com rapidez [25].

Chebotko e colegas [29] apresentam o RDFProv, um outro projeto de pesquisa que utiliza os recursos de Cloud Computing para aperfeiçoar workflows científicos. RDFProv é definido como um armazenamento de RDFs de forma relacional, que é otimizado para consultas e gerenciamento de dados de proveniência para workflows científicos. Além disso, sua abordagem utiliza as capacidades de interoperabilidade, extensiabilidade e inferência providas pelas tecnologias da web semântica. Desta forma, RDFProv é capaz de juntar as habilidades de armazenamento e consultas de um RDBMS para ajudar a atender os requisitos inerentes ao problema de gerenciamento de dados de proveniência em workflows científicos.

Diferentemente da abordagem de Newman e colegas, nossa implementação não utiliza uma nova forma de estrutura de triplas com a finalidade de facilitar a integração de dados entre alguns banco de dados público e o ProteinWorldDB [30]. Nosso trabalho é semelhante ao de Husain e colegas [25] na utilização do Hadoop para processamento distribuído, não para armazenar ou recuperar triplas RDF, mas ao invés disso, para auxiliar no processo de mapeamento dos dados brutos para a representação específica em triplas RDF, isto é no processo de "triplificação". Os dados utilizados nesse processo foram originalmente obtidos a partir do banco de dados do Protein World DB, o qual objetivo principal é possibilitar o compartilhamento do resultado das comparações das sequências de proteínas.

4 Uma Proposta de Arquitetura

O objetivo deste trabalho consiste em descrever uma arquitetura destinada a possibilitar o mapeamento dos dados do banco de dados do ProteinWorldDB em triplas RDF, com a finalidade de gerar consultas semânticas sobre esses dados mapeados. Para atingir este objetivo, nós aproveitamos o baixo custo inicial e a escalabilidade que *Cloud Computing* oferece, utilizando também o framework MapReduce para realizar o processamento desses dados. Além disso, o objetivo secundário é compartilhar essas grandes quantidades de dados para que outros projetos de pesquisam possam utilizar esses dados de uma forma fácil e ágil.

A nossa proposta de arquitetura utiliza uma infraestrutura local de baixo custo que possui somente *commodities machines* e uma infraestrutura pública de *Cloud Computing*. A nossa infraestrutura local possui limitações de hardware, devido a utilizarmos computadores pessoais, ou seja, não específicos e dedicados para o processamento de alto desempenho. Pela limitação de hardware também optamos por utilizar um sistema distribuído para o armazenamento do grande volume de dados do ProteinWorlDB. Essa infraestrutura local é utilizada também para realizar análises locais utilizando uma implementação do *framework* MapReduce, antes mesmo de podermos testar em uma nuvem pública. Assim, o objetivo desta infraestrutura local é principalmente utilizar o *framework* MapReduce para possibilitar testes de corretude dos nossos métodos de análise para, posteriormente, continuar com o processamento do grande volume de dados.

A figura 1 ilustra a nossa proposta de arquitetura para a execução dos nossos procedimentos de análise e processamento dos dados de resultado do ProteinWorldDB. Esta infraestrutura local é um tipo de *Private Cloud*, onde podemos utilizar alguma ferramenta de suporte a criação e gerenciamento de infraestruturas de *cloud computing*, tais como: *Open Eucalyptus* [31], *OpenStack* [32], e *OpenNebula* [33]. Basicamente, a nossa *private cloud* possui um repositório distribuído e um conjunto de nós de processamento, onde possamos armazenar os dados de resultado do ProteinWorldDB e também realizar alguns processamentos locais. Os conjuntos de nós devem ser utilizados juntamente com alguma implementação do *framework* MapReduce. O processamento dos dados distribuídos geram também novos dados, que devem ser distribuídos entre os nós de armazenamento do sistema de arquivos da nossa infraestrutura local. Como não temos máquinas dedicas para processamento de alto desempenho e armazenamento, a escolha da ferramenta para a construção da *cloud* privada também levou em consideração esta restrição à utilização das máquinas da infraestrutura local.

Além dessa infraestrutura local, temos também um mecanismo (script, software, entre outros) para possibilitar o envio dos dados para serem submetidos para uma *cloud* pública. *Amazon Web Service* [18], *Goole App Engine* [34] e *Microsoft Azure* [35] são alguns exemplos de provedores públicos de *Cloud Computing*.

Na nossa arquitetura, a escolha de um provedor público de *cloud computing* deve levar em consideração algumas características, tais como: a existência de um mecanismo para armazenamento dos dados da nuvem privada do laboratório, um sistema que suporte a utilização de uma implementação do modelo de computação paralelo distribuído *MapReduce*. Em específico, desejamos que os dados submetidos a nuvem pública não estejam logicamente armazenados nos mesmos nós que realizam os processamento destes dados, para podermos ter uma maior flexibilidade sobre a demanda de nós de processamento, sem termos que nos preocupar com os dados armazenados nessas máquinas. Esses dados serão

utilizados para o processamento mais pesado, ou seja, aquele que não é possível de ser realizado em nossa nuvem privada devido as nossas restrições de armazenamento.

Além de um repositório de dados, necessitamos também de uma máquina para armazenar os resultados do processamento dos dados do ProteinWorldDB. Neste trabalho em específico, necessitamos de um repositório de triplas RDF para armazenar as triplas e relações que forem geradas pela nosso procedimento de triplificação 5.1.3. Este *triple store* pode ser uma implementação local, apenas uma máquina alocada para esta tarefa, ou distribuído, utilizando alguma abordagem de processamento distribuído de triplas RDF [25; 26; 27; 29]. Para disponibilizarmos os resultados das nossas análises e também para fazermos consultas sobre os dados no formato de triplas RDF, necessitamos de uma máquina onde teremos um *front-end* com interfaces para visualizar os dados, possibilitar a realização de consultas e também para compartilhar esses dados com outros grupos de pesquisa.

O nosso processo de "triplificação" faz o mapeamento dos dados resultantes das comparações entre as sequências de mais de 400 genomas [4] para triplas RDF utilizando uma versão modificada de uma ontologia [36] para guiar o mapeamento. O procedimento foi desenvolvido para ser executado sobre *jobs MapReduce* que mapeiam as linhas de resultados das comparações para as entidades, atributos e relacionamentos da ontologia. Devido a grande quantidade de dados de resultado que estamos trabalhando, e também pela limitação da infraestrutura local para armazenamento e processamento destes dados, utilizamos a nossa infraestrutura local apenas como um ambiente de testes, para podermos ajustar o programa para que seja posteriormente executado em uma nuvem pública.

Na seção 5 abordaremos maiores detalhes da implementação da nossa arquitetura, bem como especificando cada escolha de tecnologia e ferramentas utilizadas para implementar os repositórios de dados, *front-end*, a escolha da tecnologia para a infraestrutura de nós de processamento, as especificações das máquinas utilizadas em nossa infraestrutura local de armazenamento e processamento, e também descrevermos em detalhes a implementação do nosso processo de "triplificação".

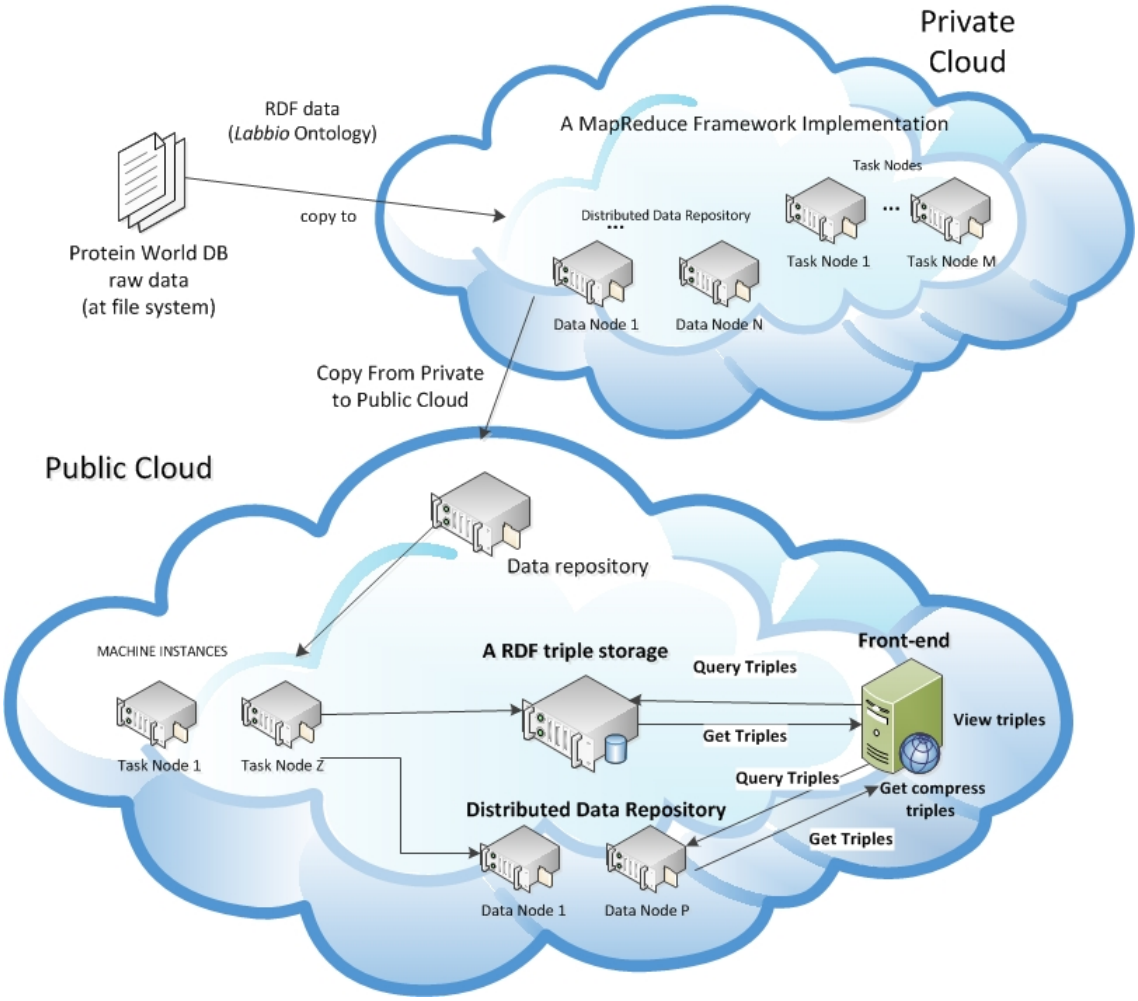

Figura 1: Arquitetura proposta.

5 Implementação da Arquitetura

Nesta seção abordaremos detalhes de implementação e instanciação da arquitetura proposta na seção 4. Em específico, descreveremos detalhes da infraestrutura local utilizada para armazenar e processar os dados de resultados do projeto de comparação genômica [4]. Descreveremos o procedimento de mapeamento dos dados de resultado para triplas RDF, possibilitando a criação de um repositório de triplas, e utilizando um vocabulário específico do domínio para permitir a integração com outras fontes públicas de dados de bioinformática.

5.1 Instanciação da Arquitetura

A nossa proposta de arquitetura utiliza duas nuvens, uma privada e uma pública, para realizar o processamento e armazenamento dos dados.

5.1.1 Nuvem privada

A nuvem privada é composta por máquinas de uso pessoal dos alunos do laboratório e desta forma, a nossa instanciação levou em consideração também a restrição de utilização desses computadores, assim como a limitação de hardware. A *cloud* privada do laboratório de pesquisa em bioinformática consiste de um ambiente heterogêneo de máquinas pessoais, composta por oito computadores, sendo 4 máquinas com processador Intel Core 2 Duo, com 2 GB de memória em média e com 300 Gb de HD. As outras 4 máquinas são equipadas com processadores Intel Pentium-IV com 1 GB de RAM e com 80 GB de HD. Devido e essas configurações heterogêneas de máquinas e também por serem máquinas destinadas ao uso pessoal dos alunos do laboratório, escolhemos utilizar neste "cluster" o Hadoop, a implementação *open-source* do modelo de computação paralelo distribuído MapReduce.

A escolha de um "cluster" Hadoop atende a nossa demanda de processamento e armazenamento local, além também de ser flexível e possível de ser utilizado levando-se em consideração a nossa limitação de hardware, permitindo assim a execução de algumas análises locais sobre o volume de dados de resultados do ProteinWorldDB. A utilização do Hadoop como uma ferramenta para a execução de tarefas distribuídas e também para o armazenamento dos dados possibilitou termos um ambiente de computação transparente, com capacidade de memória aproximadamente de 10GB e com 1.7 TB de capacidade de armazenamento. Nesta arquitetura local utilizamos o SO Debian Lenny, com a versão 0.20.2 do Hadoop e com OpenJDK java. Para monitorarmos a utilização dos recursos computacionais locais utilizamos o sistema de monitoramento do Ganglia [37].

5.1.2 Nuvem pública

Entre os provedores públicos de *cloud computing* citados anteriormente, acreditamos que o provedor da Amazon [18; 38] oferece um modelo de negócios que consideramos mais adequado em nosso trabalho [39], representando uma possível solução para uma arquitetura elástica (expansível) para a disponibilidade rápida, em tempo real, das informações dos resultados da integração dos dados do ProteinWorldDB com outros bancos de dados públicos, mesmo com o rítmo rápido de sequenciamento de novas sequências genômicas. A própria Amazon disponibiliza publicamente e sem custo algumas fontes de dados, tais como: os dados de anotação do genoma humano providos pelo projeto Ensembl [40], o conjunto de sequências de transcritos de genes bem caracterizados e de centenas de ESTs , além de informações de proteínas similares, cDNA, localização genômica dos genes, entre outros dados [41]. Além dessas fontes de dados, temos também diversas aplicações de bioinformática (*CloudBLAST, CloudBurst, Crossbow*, entre outras) já implementadas e sendo disponibilizadas através de imagens de máquinas virtuais na própria Amazon [42].

Na *cloud* pública da Amazon, utilizamos como repositório de dados o serviço de armazenamento de dados S3, que disponibiliza uma interface aos serviços web utilizados para armazenamento e obtenção de quaisquer quantidade de dados. A página web do serviço da Amazon S3 [43] traz maiores informações sobre as funcionalidades, os custos e a forma de cobrança do serviço. Utilizamos esse repositório para armazenar ambos os dados vindos da nossa nuvem privada e também dos dados gerados a partir da análise e processamento desses mesmos dados. Em nossa procedimento de triplificação, também são gerados dados intermediários, triplas no formato RDF, que também são armazenadas temporariamente no S3. Para realizar o procedimento de triplificação destes dados, utilizamos o serviço *Amazon Elastic MapReduce*, que possibilita facilmente a análise de vastos volumes de dados através da utilização do framework Hadoop, que está previamente instalado e configurado nas máquinas na *Amazon EC2* e também no serviço de armazenamento S3. Na realidade, os *jobs MapReduce* utilizam o serviço S3 como repositório para troca e geração dos dados intermediários durante o processamento dos *jobs*.

Para realizar o armazenamento das triplas e disponibilização desses dados, optamos por utilizar o servidor OpenLinkVirtuoso [44], não só por causa da robustez com relação ao gerenciamento do banco de dados oferecidos (XML e RDF), mas também pelas vantagens das tecnologias da web semântica, como a capacidade de inferência, a definição de esquemas de integração de dados, SPARQL endpoints [45], servidor WebDAV, plataforma de web services para SOA, um servidor web com as aplicações phpBB3, Drupal, WordPress, MediaWiki, entre outros serviços oferecidos [46]. Habilitar o acesso aos dados através de um SPARQL endpoint, fornece um instrumento a mais para os pesquisadores realizarem consultas semânticas no repositório de triplas RDF, trazendo novas oportunidades para a compreensão destas grandes quantidades de dados.

Nesta abordagem, temos tanto o *front-end* quanto o servidor de triplas sendo executados em apenas uma máquina. No entanto, sabemos que podemos ao invés de utilizar essa abordagem, utilizarmos o próprio serviço S3 para armazenar as triplas no formato RDF e desta forma, utilizar a abordagem de Husain e colegas [25] para recuperar as triplas. Para simplificarmos e também testarmos a nossa proposta de arquitetura, utilizamos a abordagem de um servidor de *triple store*: VirtuosoDB, Sesame, entre outros [47], com uma interface SPARQL Endpoint para permitir consultas SPARQL sobre os dados. A figura 2 ilustra a nossa implementação da arquitetura apresentando as tecnologias utilizadas.

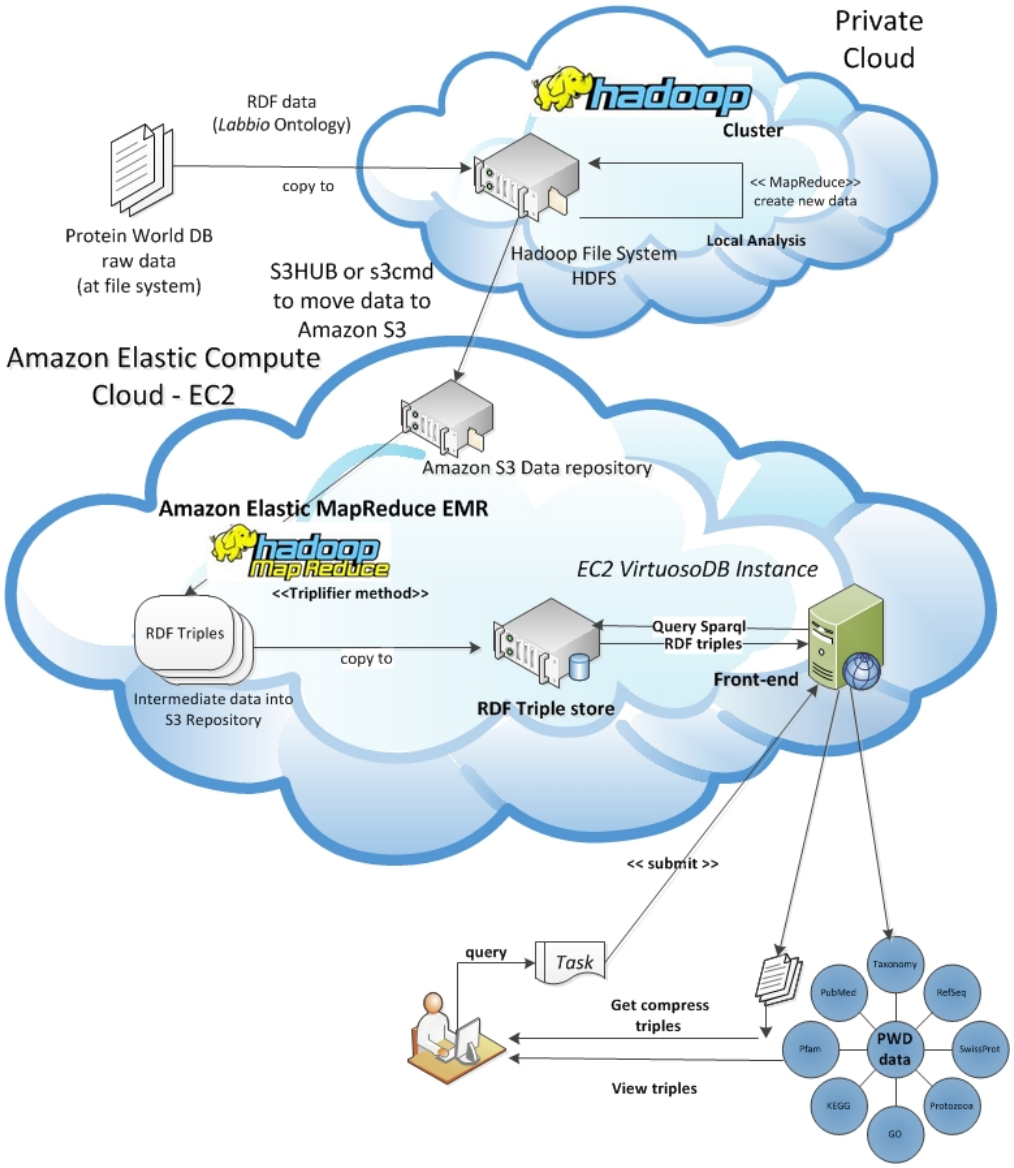

Figura 2: Implementação da arquitetura proposta.

5.1.3 Procedimento de triplificação

O Departamento de informática da PUC-Rio possui projetos de pesquisa juntamente com o instituto Oswaldo Cruz (FIOCRUZ) para investigar estratégias eficientes e inovadoras para o gerenciamento de dados biológicos. O projeto de pesquisa de comparação de genomas (GCP) [4] executou a comparação para-a-par entre todas as sequências protéicas preditas de mais de 400 genomas envolvidos, resultando num conjunto de índices de similaridades que juntamente com a nomenclatura padronizada de genes e seus produtos, possibilitou a construção de um repositório de dados de referência para os biólogos. Os dados resultantes das comparações consistem em 1.822.000 arquivos de saída de extensão "OUT", que foram distribuídos compactados e empacotados em 911 arquivos TAR. Este conjunto de dados

está armazenado localmente na nuvem privada do laboratório de pesquisa de bioinformática da PUC-Rio, e desempacotados somam aproximadamente mais de um terabyte de dados, o que mostra o dilúvio de dados que a área de bioinformática está trabalhando.

A figura 3 ilustra um trecho do arquivo de resultado **100000001-10001869.out**. Cada linha de um arquivo de saída contém os seguintes dados: identificadores da sequências comparadas, tamanho do alinhamento entre as sequências, coordenadas das regiões mais similares, porcentagem de identidade do alinhamento, números de *gaps* introduzidos nas sequências de consulta e busca durante o alinhamento, a pontuação obtida para o alinhamento entre as duas sequências, e o valor esperado ou $E - value$. Esses dados foram inicialmente armazenados em um banco de dados relacional DB2 e cada par de sequências definidas pelos identificadores das duas sequências comparadas foi denominado por *hit*.

```
67904470,900000000180997,91,30.1,0.41,0.208,226,387,601,5,211,11,19
67515963,900000000180726,79,28.5,0.29,0.212,113,276,370,21,132,18,1
67517698,900000000180389,69,27.3,0.55,0.313,48,338,382,35,82,3,0
67517698,900000000180388,69,27.3,0.55,0.313,48,338,382,35,82,3,0
67521660,900000000181120,84,27.0,0.58,0.306,72,280,350,1,72,1,0
67522172,900000000180836,61,24.9,0.98,0.529,17,112,128,1,17,0,0
67523197,900000000181782,88,27.7,0.53,0.278,79,147,218,9,87,7,0
```

Figura 3: Trecho do arquivo de resultado 100000001-10001869.out.

Para possibilitar a integração semântica dos dados dos resultados do ProteinWorlDB com outras fontes de dados, devemos inicialmente mapeá-los para o modelo de dados de triplas RDF. Neste caso, utilizamos uma ontologia para guiar o mapeamento. A ontologia utilizada no mapeamento é similar em tese ao do trabalho do SSWP [36], com algumas pequenas modificações, para permitir a reutilização e adaptação dos conceitos definidos anteriormente. Esta ontologia é similar a SSWP pois descreve as proteínas e suas propriedades, além dos relacionamentos de similaridades (*hits*) contidos nos arquivos de resultados gerados pelo projeto GCP [4]. O mapeamento entre os dados de resultados para o modelo de dados RDF denominamos por **processo de triplificação**.

A figura 4 ilustra a ontologia utilizada para realizarmos o mapeamento dos dados de resultados. Esta ontologia foi utilizada para possibilitar que os dados mapeados possam ser utilizados como um passo de pré-processamento para algumas abordagens de genômica comparativa, as quais não poderiam tirar proveito do uso desses dados em suas análises, sem antes serem mapeamentos semanticamente.

Figura 4: Ontologia utilizada no mapeamento.

Nesta ontologia criamos 5 classes de interesse: *HighInteresting*, *LowInteresting*, *MediumInteresting*, *NoInteresting* e *TopInteresting*; para classificar as proteínas conforme um valor de similaridade que possuem com outra proteína. Neste caso, utilizamos o valor de $E - value$ como parâmetro de similaridade. A figura 5 ilustra um exemplo de definição de uma regra para a classe *HighInteresting*. Nestes caso, temos uma regra diferente para cada classe de interesse.

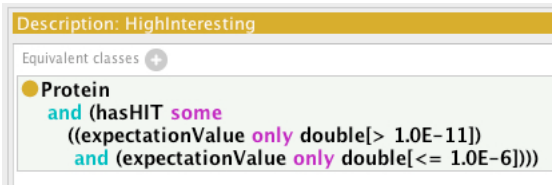

Figura 5: Definição de classe equivalente *HighInteresting*.

Como as regras estão definidas na própria ontologia, poderíamos utilizar os dados de resultados no formato de triplas RDF juntamente com essas regras de inferência previamente habilitadas, como um passo de pré-processamento de clusterização de outras metodologias de genômica comparativa, como a metodologia do 3GC [48; 49]. Desta forma, os cientistas poderíamos criar tantas regras quando acharmos necessárias e aplicá-las ao método 3GC para tentar descobrir ou predizer padrões nos dados sendo analisados.

O nosso processo de "triplificação" consistiu de *jobs MapReduce* que mapeiam cada linha nos atributos da entidade **Hit** da nossa ontologia. Desta forma, para toda linha dos arquivos resultantes de comparação são geradas 17 triplas de dados, as quais descrevem o relacionamento entre as proteínas e o resultado das suas comparações. Além disso, esses triplas também estão relacionadas com fontes de dados externas, tais como UniProt [50] entre outras, pois definimos outros mapeamentos dentro da própria ontologia para essas fontes de dados. A figura 6 ilustra um trecho do arquivo part-r-0000 referente ao resultado do mapamento dos dados de um arquivo *.out para o modelo de dados RDF. Neste trecho de arquivo podemos observar a descrição da ocorrência de dois *hits* com seus *data properties*. No final deste trecho temos a definição de uma proteína e a sua referência para a base externa do Bio2RDF [51].

```
<OntologyLabbio:hit108756783-17549739> rdf:type <OntologyLabbio:HIT> ;
    OntologyLabbio:swScore "92" ;
    OntologyLabbio:bitScore "28.9" ;
    OntologyLabbio:percentIdentity "0.519" ;
    OntologyLabbio:expectationValue "0.22" ;
    OntologyLabbio:alignLength "27" ;
    OntologyLabbio:queryStart "166" ;
    OntologyLabbio:queryEnd "192" ;
    OntologyLabbio:subjectStart "205" ;
    OntologyLabbio:subjectEnd "230" ;
    OntologyLabbio:queryGaps "0" ;
    OntologyLabbio:subjectGaps "1" .
<OntologyLabbio:hit108756783-50841531> rdf:type <OntologyLabbio:HIT> ;
    OntologyLabbio:swScore "86" ;
    OntologyLabbio:bitScore "27.6" ;
    OntologyLabbio:percentIdentity "0.250" ;
    OntologyLabbio:expectationValue "0.67" ;
    OntologyLabbio:alignLength "120" ;
    OntologyLabbio:queryStart "9" ;
    OntologyLabbio:queryEnd "126" ;
    OntologyLabbio:subjectStart "144" ;
    OntologyLabbio:subjectEnd "256" ;
    OntologyLabbio:queryGaps "2" ;
    OntologyLabbio:subjectGaps "7" .
 <OntologyLabbio:78067928> rdf:type <OntologyLabbio:Protein> ;
   OntologyLabbio:hasHIT  <OntologyLabbio:hit59801089-78067928> ;
   owl:sameAs <bio2rdfgi:78067928> .
```

Figura 6: Trecho do arquivo part-r-0000 de resultado do mapeamento para o modelo de dados RDF.

Devido a grande quantidade de dados gerada pelo projeto GCP, e também a limitação da infraestrutura local, decidimos primeiramente executar o nosso processo de triplificação primeiro localmente para obtermos os tempos de execução e estimativas de custo. Dessa forma, executamos o processo em nosso próprio *cluster Hadoop*, e em seguida no que é fornecido pela Amazon [38].

5.2 Resultados

Nas seções seguintes descreveremos maiores detalhes relacionados a escolha da quantidade de dados para serem executados na nuvem privada, a escolha de alguns parâmetros de configuração para executar os *jobs MapReduce* além de mostrarmos os tempos de execução do processo de triplificação na nossa nuvem privada e as estimativas de execução na nuvem

pública da Amazon.

Antes de executarmos quaisquer procedimentos em uma *cloud* seja ela privada ou pública, devemos primeiramente escolher alguns valores para determinados parâmetros utilizados pelo Hadoop ou pelo *Elastic MapReduce framework (EMR)*, o *framework* que possibilita o processamento paralelo dentro da cloud da Amazon (EC2), tais como: o número máximo de *map tasks*, arquivos para serem processados, nome do arquivo do programa *mapper*, nome do arquivo do programa *reducer*, *path* de saída dos resultados do processamento, entre outros parâmetros. Wall e colegas [14] apresentam um trabalho descrevendo alguns dos principais parâmetros necessários a execução de *jobs MapReduce* na Amazon.

5.2.1 Configuração da *cloud* privada

Na nuvem privada do laboratório utilizamos o Hadoop versão 0.20.2 especificando os seguintes parâmetros de configuração:

```
mapred.reduce.tasks = 3
mapred.job.reuse.jvm.num.tasks = 3;
```

Devido a nossa limitação de memória no cluster local, estamos especificando o número máximo de tarefas que o Hadoop deve reutilizar, além do número máximo de *reduce tasks* que devem ser executadas. Na execução do processo de triplificação não estamos utilizando o módulo de *streaming* do Hadoop. Desta forma, passamos para o próprio programa triplificador o diretório de entrada dos arquivos de extensão "OUT" (*triplify-input*) e também o caminho do diretório de saída no HDFS para armazenar os arquivos de saída do *job MapReduce* (*triplify-output*). Utilizamos os valores padrões para os parâmetros restantes de configuração do Hadoop. A documentação do Hadoop traz maiores detalhes sobre valores para alguns parâmetros de configuração [52] de um *cluster* Hadoop.

5.2.2 Processamento na *cloud* privada

Devido a limitação de hardware da nossa nuvem privada, tanto para processamento, quanto para armazenamento, executamos o procedimento de triplificação para um arquivo **.TAR* contendo cerca de 1.71 GB de dados de resultados descompactados.

Durante os testes iniciais utilizando as 8 máquinas do laboratório, percebemos que as máquinas com a configuração inferior, Pentium-IV e com 1 GB de RAM, demoravam muito para terminar a execução dos seus *jobs*, sendo muitas vezes colocadas na *blacklist* de máquinas que falharam do *Hadoop*. Desta forma, limitamos os nossos testes as outras 4 máquinas do "cluster" (Core 2 Duo com 3 GB de RAM). Nesse ambiente, os testes utilizando os arquivos de resultado ***.out** de apenas um arquivo empacotado TAR, foram executados aproximadamente durante 4 horas e 40 minutos (4h 39' 38"), gerando um conjunto de mais de 12 GB de dados de triplas RDF.

No nosso programa de triplificação disparamos um *job MapReduce* para todos os arquivos **.out* contidos no diretório de entrada passado como parâmetro. Desta forma, o *framework Hadoop* dispara M *jobs mapper*, um para cada arquivo de entrada e R *jobs reduce*, onde R é o valor definido estaticamente nos arquivos de configuração do Hadoop pelo parâmetro *mapred.reduce.tasks*. Assim, podemos calcular o número de tarefas a

serem executadas no cluster com a seguinte fórmula: $N_{TAR} * (M + R)$, onde N_{TAR} corresponde ao número de arquivos TAR que serão processados, M ao número de arquivos OUT existentes e R é um parâmetro. Como cada arquivo TAR possui 2000 arquivos OUT compactados, temos que o número de tarefas que devem ser executadas é dado por: $numero\_tarefas = N_{TAR} * (2000 + R)$. Esta fórmula é útil para ajudar na avaliação da escolha da instância de máquina que deverá ser utilizada nos testes da *cloud* pública da Amazon.

5.2.3 Configuração da *cloud* pública

Na nuvem da Amazon utilizamos os parâmetros de configuração padrão do Hadoop tanto para o sistema de arquivos do HDFS, quanto para as variáveis de configuração para os *jobs MapReduce*. Como estamos utilizando nos testes as instâncias do tipo *large* temos como configuração padrão a execução de 4 *mappers* e *reducers* por máquina instanciada. Para os testes de estimativas de execução, utilizamos 4, 8 e 16 máquinas. Desejávamos testar em mais máquinas, mas para isso seria preciso fazer uma solicitação formal a equipe da Amazon, descrevendo o motivo do aumenta da demanda. Como queríamos apenas fazer testes preliminares, restringimos a utilização a no máximo 20 máquinas.

5.2.4 Processamento na *cloud* pública

Comparando a configuração das máquinas da nossa infraestrutura local com alguns tipos de máquinas disponíveis na Amazon, estimamos que as nossas máquinas são equivalentes em processamento as máquinas *small* disponíveis pela Amazon. Desta forma, ao executar um teste com apenas um arquivo *.out com aproximadamente 404 KB, o tempo total de processamento foi de 3 minutos. Para realizar o processamento de apenas um arquivo *.tar contendo 2000 arquivos *.out de resultados, teríamos que realizar mais de 2000 tarefas. Pela configuração padrão da instância *small* da Amazon EC2, temos que o número máximo de *mappers* por máquina executando são 2 e apenas uma tarefa *reducer*. Como o *framework Elastic MapReduce* da Amazon determina o número de tarefas *mapper* a partir do tamanho e do número de arquivos de entrada, teríamos 1000 tarefas *mapper* para serem executadas mais uma tarefa *reducer*, para instâncias do tipo *small*. Desta forma, para processar os 2000 arquivos *.out em apenas uma máquina teríamos que gastar aproximadamente 3000 minutos, o que se torna inviável para este tipo de instância. Desta forma, decidimos por utilizar instâncias do tipo *large*, que promovem tanto um custo e tempo efetivo para processarmos o conjunto de dados. Embora escolhas alternativas, tais como as instâncias do tipo *small* ou *medium* são mais econômicas por hora de processamento [53], nossos cálculos demonstraram que esses instâncias alternativas poderiam requere um tempo computacional ainda maior para completar o procedimento de triplificação, resultando em custos semelhantes aos de uma instância do tipo *large*.

Antes de transferirmos os dados de resultados do projeto GCP para serem processados na Amazon, fizemos uma estimativa de custos de armazenamento e de processamento para todo o conjunto de dados, com a finalidade de estimarmos a quantidade de recursos financeiros e também o tempo total gasto com o processamento dos dados. Para estimarmos o tempo de processamento e também para compreendermos as questões relacionadas a escalabilidade na Amazon, decidimos por testar o mesmo conjunto de dados com aproximadamente 1.71 GB, mas alternando a quantidade de máquinas utilizadas para

o processamento desses dados.

As tabelas a seguir ilustram os custos para armazenamento e processamento de todos os dados de resultados do projeto GCP [4] em triplas RDF. A tabela 1 descreve os custos de armazenamento para apenas um único arquivo TAR, e também para todo o conjunto de dados de resultados (911 arquivos TAR) utilizando o serviço de armazenamento Amazon S3 [43]. O custo de armazenamento é dividido em três categorias: **dados de entrada**, **dados de logs das operações** e **dados de saída**. Como podemos observar na tabela 1, os dados dos logs acabam sendo um fator de impacto sobre o custo total. Não considerando o fato de que o processo de geração de logs pode não ser uma parte fundamental do nosso processo, os logs representam somente a única maneira para controlar a forma como o processo de triplificação está sendo executado. Isso acontece pois o framework *MapReduce* é oferecido como um serviço [38], e os logs do *Hadoop* podem conter informações importantes, caso exista algum problema durante o processo de triplificação dos dados.

Table 1: Estimativa de custos de armazenamento na Amazon.

	Custos dos arquivos *.tar			
# Tar files	**1**		**911**	
Saída (GB)	12,7875	$1.82	11649,4125	$1.631,00
Logs (GB)	1	$0,14	911	$127,54
Entrada (GB)	1,67	$0,28	213,08	$29,96
Total	**$2,24**		**$1.788,50**	

As tabelas 2 e 3 mostram, respectivamente, o custo e tempo de processamento para apenas um arquivo *.TAR e também as estimativas de custo e processamento para todo o conjunto de dados de resultados. Uma característica interessante que podemos notar nos resultados dessas tabelas é que, embora o custo de processamento não varie muito, dentro de cada tabela específica, o tempo de processamento diminui linearmente com o aumento do número de máquinas. No entanto, o único problema com o aumento do tamanho do *cluster* é que teremos, provavelmente, que fazer uma solicitação o provedor da cloud pública para liberar mais máquinas para o processamento. Por exemplo, utilizando a Amazon [18] como a cloud pública, se necessitássemos de um *cluster* com mais do que 20 nós, teríamos que enviar uma solicitação por email explicando os motivos para alocarmos a quantidade de nós desejada.

Table 2: Tempo de processamento de um arquivo TAR na Amazon utilizando instâncias *large*.

	Tamanho do Cluster		
1(1.71 GB)	**4**	**8**	**16**
Tempo	2h 44min	44min	22min
Custo Processamento($)	$0,72	$0,48	$0,96
Custo total	**$2,96**	**$2,72**	**$3,20**

Table 3: Estimativa de tempo de processamento para 911 arquivos TAR na Amazon utilizando instâncias *large*.

	Tamanho do Cluster		
911(> 1 TB)	**4**	**8**	**16**
Tempo	1898	896	441
Custo Processamento($)	$455,52	$430,08	$423,36
Custo total	**$2.244,72**	**$2.219,28**	**$2.212,56**

5.3 Discussão

O advento das novas tecnologias de sequenciamento de DNA têm demandado continuamente soluções computacionais de novas arquiteturas e ferramentas de alto desempenho para auxiliar tanto no processo de comparação, quanto no processo de interpretação dos dados dos genomas. O trabalho que está sendo apresentado descreve uma arquitetura e uma instanciação para a execução do procedimento de mapeamento dos dados de resultados do projeto de comparação de genomas GCP para o modelo de dados RDF, utilizando-se o serviço *Elastic MapReduce* (EMR) disponibilizado pela *Amazon Elastic Compute Cloud* (EC2).

Para se desenvolver a maioria das aplicações para o ambiente de *cloud* é necessário que estas aplicações tenham que se adequar aos requerimentos impostos pelas implementações do Hadoop ou da própria Amazon EMR. Estas aplicações estão muitas vezes implementadas na linguagem Java e muitas vezes são embaraçosamente paralelas. Entretanto, muitos algoritmos e aplicações em bioinformática, especificamente em genômica comparativa, são compostos por um conjunto de programas, muitos deles escritos em outras linguagens ao invés de Java, tal como perl, C, python, Ruby, entre outras. Assim, essas aplicações e algoritmos não estão em conformidade com os requerimentos do Hadoop e EMR. Apesar disso, ambos provêm a funcionalidade de execução através do uso do módulo de *streaming*, permitindo a passagem dos dados via *standard input* diretamente para ser processado pela função *mapper*, que passaria os resultados via *standard output* para a função *reducer* realizar o seu processamento.

No presente trabalho, embora tenhamos desenvolvido nosso próprio procedimento de triplificação, poderíamos ter utilizado algum procedimento de triplificação já implemen-

tado, utilizando o módulo de *streaming*. Neste caso, o procedimento de triplificação poderia ter sido implementado em qualquer outra linguagem diferente de Java, necessitando que fizéssemos maiores alterações no processo de computação das triplas RDF. Estas alterações includem a criação dinâmica de um arquivo de entrada contendo a linha de comando do programa de triplificação e os parâmetros necessários a sua execução. Desta forma, este arquivo seria passado via *streaming*, à função *mapper* que leria os comandos linha por linha, e dispararia o procedimento de triplificação para cada arquivo específico. Neste caso, como os arquivos *.out são muito pequenos, não teríamos um ganho significativo ao distribuir a computação para cada arquivo *.out. Esta abordagem seria mais adequada para o processamento em paralelo do conjunto de dados dos 911 arquivos *.tar. Neste caso, o arquivo de entrada poderia conter como parâmetro para o procedimento de triplificação, a pasta contendo o determinado arquivo *.tar já descompactado, ou seja, a pasta onde estão os 2000 arquivos *.out referentes ao arquivo TAR específico. Assim, poderíamos escalar a computação de todo o conjuntos de dados.

Em ambas as abordagens com e sem o uso do módulo de *streaming*, temos como resultado da execução da tarefa *reducer* arquivos denominados **part-r-numero**, onde número refere-se ao número sequencial atribuído ao arquivo durante a sua geração, em relação aos demais arquivos. Estes arquivos contém os dados de resultados mapeados para o modelo de dados de RDF, levando-se em consideração a ontologia utilizada no mapeamento.

Muitas outras tarefas de bioinformática podem nem requerer a utilização de uma tarefa *reducer*, tendo em vista que o resultado obtido a partir da execução de um programa de bioinformática específico é gerado a partir da execução do programa pela tarefa de um *mapper*. Assim, poderíamos construir um fluxo de execução de tarefas, onde diferentes programas de bioinformática poderiam ser executados em pipeline utilizando os resultados gerados pelas tarefas *mapper* específicas. Esta abordagem permitiria que diversas aplicações que não foram descritas na linguagem Java pudessem diretamente serem executadas com modificações apenas nos arquivos de entrada passados para cada tarefa *mapper*. Desta forma, acreditamos que esta abordagem de *streaming* juntamente com utilização do Hadoop e EMR da Amazon, representam um alternativa interessante às abordagens ditas tradicionais de computação de alto desempenho para aplicações em bioinformática.

5.4 Materiais e métodos

Nesta seção descreveremos como fizemos para transferir os dados para a nuvem pública da Amazon para serem processados e também os requisitos do método de triplificação, dos dados, a configuração dos componentes e dos caminhos para executar o procedimento de triplificação na Amazon, como fizemos a monitoração dos serviços que eram executados e também descreveremos sobre a portabilidade da nossa aplicação.

5.4.1 Serviços requisitados na Amazon

Para utilizar os serviços na Amazon primeiramente devemos criar uma conta que provê o acesso aos 2 serviços que foram utilizados: *Elastic MapReduce framework* (EMR) e *elastic storage unit* S3. O acesso a esses serviços foi garantido através da utilização do sistema de chave pública e privada. O custo por hora de utilização de apenas uma instância de máquina do tipo *large* utilizando o serviço *Elastic MapReduce* foi de $0.06. O custo de armazenamento utilizando o serviço S3 foi de $0.140 por GB armazenado, $0.0 por GB

transferido e de $0.01 para cada 1000 requisições de transferência de saída da nuvem. Não tivemos gastos para transferir os dados para a nuvem devido ao período de *free-transfer*, onde as transferências para a nuvem não eram cobradas até o primeiro TB de dados. Atualmente, o custo para transferir os dados para a nuvem é de $0.10 por GB transferido para o primeiro TB de dados. Maiores detalhes sobre os custos de armazenamento do serviço S3 estão disponíveis em [54].

5.4.2 Necessidades do método de triplificação

O método de triplificação consiste em apenas um arquivo **jar** (*triplify-mapred.jar*). Este arquivo foi transferido para o nosso repositório no serviço S3 denominado por *labbio.pwd.triplify* na pasta *executables*. Como as bibliotecas utilizadas para compilar o código fonte já estão todas compactadas no arquivo jar, não foi necessário transferir mais dados para podermos executar o procedimento de triplificação. Os dados de entrada para executar o procedimento devem estar armazenados no serviço de armazenamento da Amazon S3.

5.4.3 Requerimentos dos dados

Os dados utilizados nos testes de custo do procedimento de triplificação foram obtidos a partir do arquivo **10001-12000_results.tar**, selecionado da lista dos 911 arquivos disponíveis. Este arquivo foi desempacotado e resultou em 2000 arquivos compactados de resultados, referenciando a comparação entre todos os blocos de sequências de 1 à 2000. Esses arquivos foram descompactados localmente na nossa nuvem privada utilizando-se um *script* em *bash*. Uma vez descompactados, todos os 2000 arquivos foram transferidos para o *bucket labbio.pwd.triply* e armazenados na pasta *input*. Como estávamos fazendo apenas testes de custo não foi preciso utilizar uma outra estratégia para a transferência desses dados. No entanto, para realizarmos o processamento de todo o conjunto dos 911 arquivos, teremos ainda que implementar uma outra estratégia para transferir os dados mesmo compactados, em demanda, e utilizarmos uma outra máquina para apenas descompactar e salvar os arquivos **.out* na pasta *input*.

5.4.4 Configuração dos componentes e pasta de resultados na Amazon

Para transferirmos os dados e o programa de triplificação da nuvem privada para a nuvem pública da Amazon, utilizamos o programa cliente S3HUB [55] para o serviço Amazon S3. Na web existem diversas aplicações [56; 57] semelhantes para auxiliar graficamente no gerenciamento dos dados no serviço S3 da Amazon. Embora utilizarmos esta ferramenta gráfica para nos auxiliar no procedimento de transferência dos dados, poderíamos também ter utilizado alguma ferramenta *open-source* de linha de comando, tal como o cliente *s3 command client* [58]. Esta ferramenta possibilita a transferência e o gerenciamento dos dados dentro da própria Amazon S3. Através da utilização desta ferramenta poderíamos criar *buckets* para transferir os dados para o serviço S3 especificando um simples argumento **mb**. Desta forma, para transferir os arquivos *.out para o *buckect labbio.pwd.triplify*, teríamos que executar **s3cmd put *.out s3://labbio.pwd.triplify/input/**. A Amazon disponibiliza um conjunto de ferramentas para auxiliar no processo de configuração e gerenciamento dos seus serviços [59].

5.4.5 Configuração do *job MapReduce*

Para executarmos o procedimento de triplificação alguns ajustes nos parâmetros de configuração para os *jobs MapReduce* podem ser passados via linha de comando, utilizando-se alguma ferramenta de linha de comando, tal como a *Amazon Elastic MapReduce Ruby Client* (Ruby CLC) [60], ou através da própria interface web AWS Management Console, durante o passo de criação de um fluxo de trabalho. Embora saibamos a importância da configuração de valores adequados para determinados parâmetros, neste trabalho como as tarefas são processadas rapidamente e também não fazem muito uso de recurso de memória, decidimos por utilizar os valores de parâmetros padrões para estes parâmetros.

Em específico, poderíamos ter configurado um valor para o parâmetro *mapred.task.timeout*. Este parâmetro especifica o numero máximo de horas necessárias para executar uma tarefa *MapReduce*. Se este valor foi ajustado para um valor muito baixo, as tarefas que excedem este limite superior são anotadas pelo programa *Elastic MapReduce* (EMR) como tarefas que falharam, e após quarto tentativas sem sucesso, a máquina que executou a tarefa é colocada na *blacklist* do EMR, assim como acontece no Hadoop, e não estarão mais disponíveis para as próximas execuções. Para uma aplicação que requer muitas horas de processamento, faz-se necessário estabelecer um valor adequado para este parâmetro.

Um outro parâmetro interessante para ser avaliado é o *java heap space*. Caso uma aplicação necessite de uma capacidade de memória maior do que o valor padrão, o *daemon JobTracker* poderia frequentemente receber uma mensagem de falta de memória, requisitando uma reinicialização manual do seu serviço. Utilizando-se o cliente *Ruby CLC* [60] é possível alterar essa configuração para cada tipo de tarefa que desejarmos executar. Desta forma, se um determinado *job* for composto por tarefas com requisições de memória diferentes, podemos disparar cada tarefa com um valor específico, evitando falhas durante o processamento de cada tarefa especificamente.

5.4.6 Monitoração da execução do procedimento

Para possibilitar o monitoramento e o *debug* das tarefas, habilitamos a geração dos logs antes da execução das tarefas de triplificação, para permitir acompanhar o processamento dos dados e também para termos uma fonte de recursos a recorrer, caso algum erro ocorra durante o procedimento de triplificação. Com os *logs* habilitados podemos visualizar o acompanhamento da execução de cada tarefa, assim como os arquivos lidos e escritos de cada máquina. No entanto, a geração de *logs* também tem um impacto sobre o custo de execução das tarefas, uma vez que os arquivos de *logs* precisam ser armazenados no serviço de armazenamento S3. Para executar os nosso testes de custo, criamos o diretório *logs* para armazenar os arquivos. Durante nossos testes não fizemos a remoção dos arquivos de *logs* para cada teste, deixando esta tarefa para o final do conjunto de testes como um todo. Isso gerou uma demanda de armazenamento muito grande, além de dificultar o processo de remoção destes arquivos no final dos testes, uma vez que são gerados muitos arquivos para cada teste. Assim, sugerimos para as próximas execuções de tarefas, a execução de um *script* que faça a remoção automática desses *logs* quando a tarefa executar com sucesso.

5.4.7 Portabilidade

Neste trabalho fizemos uma maior utilização da *cloud* pública da Amazon EMR para realizar o processamento de parte do conjunto de dados de resultados do projeto de com-

paração de genomas GCP. Embora tenhamos escolhido este provedor específico de *cloud computing*, temos que o nosso procedimento de triplificação pode ser executado utilizando apenas a implementação Hadoop do *framework MapReduce*, com praticamente pequenas mudanças relacionadas aos parâmetros de execução e instanciação da arquitetura. Como fizemos os testes iniciais na nossa própria nuvem privada utilizando apenas o Hadoop, o processamento de todo o conjunto de dados poderia, em teoria, ser realizada na nossa própria nuvem privada, se tivéssemos capacidade de armazenamento e processamento para realizar esta tarefa rapidamente. Mesmo utilizando-se um *cluster* Hadoop, poderíamos ter *scripts* pré-configurados para instanciar o nosso ambiente da cloud privada, da mesma forma como a Amazon disponibiliza através de seu console de gerenciamento.

Sugerimos que na utilização de uma cloud privada, o parâmetro de *mapred.map.tasks.speculative.execution* fosse ajustado para o valor *false*, não permitindo que o Hadoop "especule" que uma tarefa que está sendo executada por muito tempo, esteja em um estado de mal funcionamento e, desta forma, o Hadoop tome a decisão de executar a mesma tarefa em paralelo, o que muitas vezes pode degradar o desempenho nesse ambiente distribuído. Assim, como temos que o Hadoop é uma implementação *open-source* do *framework MapReduce* e que é de fácil instalação em um ambiente Linux, temos então que a nossa arquitetura, bem como a utilização do procedimento de triplificação, pode ser instanciada e executado sobre um ambiente de *cloud computing* que utilize o SO Linux.

6 Trabalhos Futuros

Nesta seção abordaremos as nossas sugestões para trabalhos futuros. Como uma primeira sugestão citamos a necessidade de ajustarmos o nosso processo de instanciação do ambiente da nuvem na Amazon. Neste trabalho fizemos a configuração da escolha do tipo de máquinas, escolha de parâmetros, além da quantidade de máquinas, entre outros parâmetros, utilizando os recursos gráficos da própria interface de gerenciamento da Amazon. Para deixarmos esta arquitetura mais dinâmica e de forma a ser reproduzida mais facilmente, poderíamos fazer este procedimento de instanciação de forma programática, ou seja, utilizando uma ferramenta de configuração via linha de comando, como a Ruby CLC [60]. Desta forma, bastaríamos construir *scripts* de instanciação e executá-los para diferentes parâmetros de entrada, possibilitando um melhor controle e dinamismo do processo de instanciação do ambiente computacional.

A utilização de um sistema de armazenamento de triplas RDF diferente da arquitetura proposta é também uma outra abordagem como trabalho futuro. Neste caso, poderíamos utilizar o próprio serviço de armazenamento da Amazon S3 e implementar as consultas como *jobs MapReduce* conforme a abordagem proposta por Husain e colegas [25]. Desta forma, poderíamos ter uma arquitetura ainda mais escalável e dependente apenas de uma implementação do *framework MapReduce*.

Uma outra proposta de trabalho futuro consiste na utilização de um outro *front-end* para permitir a visualização dos dados de resultados no formato RDF.

Sugerimos também a utilização de outras metodologias que poderiam fazer uso do mecanismo de inferência como um procedimento de pré-processamento ou de agrupamento dos dados em relação à definição de um conjunto de regras na ontologia. No caso da metodologia do 3GC [49] poderíamos ter a formação dos grupos de uma, duas e três sequências realizada durante a carga ou geração das triplas no formato RDF. Assim, o algoritmo de comparação de três genomas poderia fazer uso destes grupos diretamente, sem ter que computar quais as sequências fazem parte de cada grupo.

Uma outra sugestão de trabalho futuro é a adaptação do procedimento de triplificação para utilizar a especificação da W3C da linguagem de mapeamento R2RML [61], de bases de dados no modelo relacional para repositório de dados RDF. Esses mapeamentos provêm a habilidade de visualizarmos os dados organizados no modelo relacional no modelo de dados RDF, expressados na estrutura e no vocabulário de escolha do autor do mapeamento. Neste caso, o trabalho futuro poderia se concentrar na utilização desta linguagem através da execução de *jobs MapReduce*. Desta forma, poderíamos realizar o procedimento de triplificação de forma distribuída, processando tabelas e relacionamentos de forma paralela, acessando uma mesma base de dados. Esta abordagem também poderia ser útil para bases de dados No-SQL, onde os dados não estão organizadas em um modelo relacional. Mas neste caso, teríamos que rever a definição proposta pela W3C e fazer as modificações necessárias para realizar os mapeamentos dos dados.

Esperamos que esses trabalhos futuros possam influenciar mais e mais cientististas a desenvolverem e direcionarem suas pesquisas ao aprimoramento das aplicações, a descoberta de valores mais adequados para os parâmetros de configuração para cada tipo de aplicação, a uma utilização mais intensa dos recursos computacionais da *cloud* a fim de intensificar os estudos relacionados, tanto as necessidades de avanços relacionados a criação de novos algoritmos para montagem de genomas e transcriptomas, quanto aos novos tipos de estudo de *genome wide association studies* - GWAS e *phenome-wide association scans* - PheWas para

possibilitar a associação de modificações genéticas com dados de traços fenotípicos minerados a partir de registros médicos eletrônicos, onde são necessários armazenar e processar diversos tipos de registros de pacientes [13], ou seja, exigindo cada vez mais capacidade de armazenamento e processamento rápido desses dados.

7 Conclusão

Grandes volumes de dados estão sendo gerados todos os dias, e os desafios que essa gigantesca massa de dados apresenta para a comunidade de pesquisa são diferentes e demandam diferentes tratamentos para cada área específica de pesquisa. Desta forma, este trabalho propõe uma arquitetura para processamento, e compartilhamento de dados biológicos, mas também colocando à disposição a capacidade de inferência sobre essa arquitetura.

Assim, decidimos por utilizar tecnologias conhecidas e atuais para atender a objetivos importantes: primeiramente o tratamento de grandes quantidades de dados biológicos em larga escala, e em segundo, permitir a integração de dados em larga escala biológica. O nosso trabalho futuro consiste em oferecer novas mecanismos de acesso eficientes com relação a custo e tempo a essa grande quantidade de dados, através de tecnologias da Web Semântica e suas aplicações.

Apesar do fato de que nossa ferramenta de triplificação aproveita o paralelismo dado pelo framework *MapReduce* estamos cientes da falta da flexibilidade devido ao tipo específico de codificação para este problema. O mapeamento dos dados está descrito diretamente no código. Consequentemente, o trabalho futuro para utilização da linguagem de mapeamento R2RML possibilitará uma maior flexibilidade, possibilitando realizarmos o processo de triplificação para outros domínios do conhecimento e de uma forma escalável.

Ao mesmo tempo, os recursos oferecidos pelo servidor do OpenlinkVirtuoso [44] tais como SPARQL endpoints, uma máquina de inferência, entre outros [46], sabemos que não é uma abordagem completamente escalável. Isto ocorre porque o sistema irá escalar enquanto o servidor será capaz de ser ampliado (*scaling up*), mas a escalabilidade verdadeira é atingida quando pudermos alugar mais instâncias de processamento (*scaling out*) em algum provedor de *Cloud Computing*. Desta forma, estamos atualmente pesquisando variações da nossa arquitetura para cumprir as exigências do processamento e armazenamento em larga escala, para possibilitar que mais aplicações do domínio da bioinformática possam ser executadas sobre um ambiente de *Cloud Computing*.

Referências Bibliográficas

[1] Gibas C. and Jambeck P. *Developing Bioinformatics Computer Skills*. O'Reilly Media, April 2001.

[2] Heng Li and Nils Homer. A survey of sequence alignment algorithms for next-generation sequencing. *Briefings in Bioinformatics;*, 11(5):473–483, 2010. ISSN 1477-4054. doi: 10.1093/bib/bbq015.

[3] Thomas D. Otto, Marcos Catanho, Cristian Tristao, Marcia Bezerra, Renan M. Fernandes, Guilherme S. Elias, Alexandre C. Scaglia, Bill Bovermann, Viktors Berstis, Sergio Lifschitz, Antonio B. de Miranda, and Wim Degrave. ProteinWorldDB: querying radical pairwise alignments among protein sets from complete genomes. *Bioinformatics*, 26(5):705–707, March 2010. doi: 10.1093/bioinformatics/btq011. URL `http://dx.doi.org/10.1093/bioinformatics/btq011`.

[4] Genome Comparison Project - GCP. Disponível em URL: `http://www.dbbm.fiocruz.br/labwim/bioinfoteam/index.pl?action=gencomp`. Acesso em 5 de Janeiro de 2011.

[5] Daniel C. Koboldt, Li Ding, Elaine R. Mardis, and Richard K. Wilson. Challenges of sequencing human genomes. *Briefings in Bioinformatics*, 11(5):484–498, 2010. doi: 10.1093/bib/bbq016. URL `http://bib.oxfordjournals.org/content/11/5/484.abstract`.

[6] Protein Families - Pfam. Acesso em 05 de Janeiro de 2011. URL `ftp://ftp.sanger.ac.uk/pub/databases/Pfam`.

[7] Amos Bairoch and Rolf Apweiler. The SWISS-PROT Protein Sequence Data Bank and Its New Supplement TREMBL. *Nucleic Acids Research*, 24(1):21–25, 1996. doi: 10.1093/nar/24.1.21. URL `http://nar.oxfordjournals.org/content/24/1/21.abstract`.

[8] Brigitte Boeckmann, Amos Bairoch, Rolf Apweiler, Marie-Claude Blatter, Anne Estreicher, Elisabeth Gasteiger, Maria J. Martin, Karine Michoud, Claire O'Donovan, Isabelle Phan, Sandrine Pilbout, and Michel Schneider. The SWISS-PROT protein knowledgebase and its supplement TrEMBL in 2003. *Nucleic Acids Research*, 31(1): 365–370, 2003. doi: 10.1093/nar/gkg095. URL `http://nar.oxfordjournals.org/content/31/1/365.abstract`.

[9] Kyoto Encyclopedia of Genes and Genomes - KEGG. Acesso em 03 de Fevereiro de 2011. URL `http://www.genome.jp/kegg/`.

[10] National Center for Biotechnology Information - NCBI. Acesso em 11 de Janeiro de 2011., . URL `http://www.ncbi.nlm.nih.gov/`.

[11] Taxonomy at National Center for Biotechnology Information - NCBI. Acesso em 12 de Janeiro de 2011., . URL `http://www.ncbi.nlm.nih.gov/guide/taxonomy/`.

[12] The Gene Ontology - GO. Acesso em 04 de Fevereiro de 2011. URL `http://www.geneontology.org/GO.downloads.database.shtml`.

[13] H Craig Mak. Trends in computational biology - 2010. *Nature Biotechnology*, 29(1): 45–45, 01 2011. URL http://dx.doi.org/10.1038/nbt.1747.

[14] Dennis Wall, Parul Kudtarkar, Vincent Fusaro, Rimma Pivovarov, Prasad Patil, and Peter Tonellato. Cloud computing for comparative genomics. *BMC Bioinformatics*, 11(1):259, 2010. ISSN 1471-2105. doi: 10.1186/1471-2105-11-259. URL http://www.biomedcentral.com/1471-2105/11/259.

[15] Eric E. Schadt, Michael D. Linderman, Jon Sorenson, Lawrence Lee, and Garry P. Nolan. Computational solutions to large-scale data management and analysis. *Nature Review Genetics*, 11(9):647–657, 09 2010. URL http://dx.doi.org/10.1038/nrg2857.

[16] 1000 Genomes - A Deep Catalog of Human Genetic Variation. Disponível em URL: www.1000genomes.org. Acessado em 10 de Janeiro de 2011.

[17] World Community Grid. Disponível em URL http://www.worldcommunitygrid.org/. Acessado em 11 de Dezembro de 2010.

[18] Amazon WebServices. Disponível em URL: http//:aws.amazon.com/. Acesso em 21 de Janeiro de 2011., .

[19] Semantic Web - W3C. Acesso em 20 de Dezembro de 2010., . URL http://www.w3.org/standards/semanticweb/.

[20] Michael C Schatz, Ben Langmead, and Steven L Salzberg. Cloud computing and the DNA data race. *Nat Biotech*, 28(7):691–693, 07 2010. URL http://dx.doi.org/10.1038/nbt0710-691.

[21] A. Newman, Yuan-Fang Li, and J. Hunter. Scalable Semantics - The Silver Lining of Cloud Computing. In *eScience, 2008. eScience '08. IEEE Fourth International Conference on*, pages 111 –118, 2008. doi: 10.1109/eScience.2008.23.

[22] Jeffrey Dean and Sanjay Ghemawat. Mapreduce: simplified data processing on large clusters. *Commun. ACM*, 51:107–113, January 2008. ISSN 0001-0782. doi: http://doi.acm.org/10.1145/1327452.1327492. URL http://doi.acm.org/10.1145/1327452.1327492.

[23] RDF Current Status. Acessado em 20 de Janeiro de 2011, . URL http://www.w3.org/standards/techs/rdf#w3c_all.

[24] SPARQL Current Status. Acessado em 10 de Fevereiro de 2011, . URL http://www.w3.org/standards/techs/sparql#w3c_all.

[25] Mohammad Farhan Husain, Pankil Doshi, Latifur Khan, and Bhavani Thuraisingham. Storage and Retrieval of Large RDF Graph Using Hadoop and MapReduce. In Martin Jaatun, Gansen Zhao, and Chunming Rong, editors, *Cloud Computing*, volume 5931 of *Lecture Notes in Computer Science*, pages 680–686. Springer Berlin / Heidelberg, 2009. URL http://dx.doi.org/10.1007/978-3-642-10665-1_72. 10.1007/978-3-642-10665-1_72.

[26] Mohammad Husain, Pankil Doshi, Latifur Khan, Syeda A. Rizvi, Murat Kantarcioglu, and Bhavani Thuraisingham. Cost-based Query Processing for Large RDF Graph Using Hadoop and MapReduce. Technical report, University of Texas at Dallas, November 2009.

[27] Mohammad Husain, Pankil Doshi, James McGlothlin, Latifur Khan, Bhavani Thuraisingham, and Murat Kantarcioglu. Efficient Query Processing for Large RDF Graphs Using Hadoop and MapReduce. Technical report, University of Texas at Dallas, November 2009.

[28] Apache Hadoop. Disponível em URL: `http://hadoop.apache.org/`. Acessado em 20 de Janeiro de 2011.

[29] Artem Chebotko, Shiyong Lu, Xubo Fei, and Farshad Fotouhi. Rdfprov: A relational rdf store for querying and managing scientific workflow provenance. *Data Knowl. Eng.*, 69:836–865, August 2010. ISSN 0169-023X. doi: http://dx.doi.org/10.1016/j.datak.2010.03.005. URL `http://dx.doi.org/10.1016/j.datak.2010.03.005`.

[30] Protein World Database. Disponível em URL `http://157.86.176.108/ProteinWorldDB/index.php`. Acessado em 03 de Outubro de 2010.

[31] Open Eucalyptus. Acessado em 30 de Janeiro de 2011, . URL `http://open.eucalyptus.com/`.

[32] OpenStack. Acessado em 28 de Janeiro de 2011., . URL `http://www.openstack.org/`.

[33] OpenNebula. Acesso em 25 de Janeiro de 2011., . URL `htpp://www.opennebula.org/`.

[34] Google AppEngine. Acesso em 10 de Janeiro de 2011. URL `https://appengine.google.com/`.

[35] Microsoft Azure. Acesso em 20 de Janeiro de 2011. URL `http://www.microsoft.com/windowsazure/`.

[36] C. J. M. Viana and Almeida A. C. SSWP - Busca semântica e Proposta de workflow no domínio da Bioinformática (Semantic search and workflow proposals - SSWP). Technical report, PUC-Rio, 2008.

[37] Ganglia Monitoring System - Web Site. Acessado em 20 de Janeiro de 2011. URL `http://ganglia.sourceforge.net/`.

[38] Amazon elastic mapreduce service. Acesso em 10 de Novembro de 2010. URL `http://aws.amazon.com/elasticmapreduce/`.

[39] Michael Armbrust, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy H. Katz, Andrew Konwinski, Gunho Lee, David A. Patterson, Ariel Rabkin, Ion Stoica, and Matei Zaharia. Above the Clouds: A Berkeley View of Cloud Computing. Technical Report UCB/EECS-2009-28, EECS Department, University of California, Berkeley, Feb 2009. URL `http://www.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-28.html`.

[40] Projeto Enembl - Web Site. Acessado em 02 de Fevereiro de 2011. URL http://www.ensembl.org/index.html.

[41] Unigene provided by the national center for biotechnology information. Acessado em 20 de Fevereiro de 2011. URL http://aws.amazon.com/datasets/2283?_encoding=UTF8&jiveRedirect=1.

[42] Bioinformatics, Genomes, EC2, and Hadoop - Amazon Web Service Blog. Acessado em 02 de Março de 2011., . URL http://aws.typepad.com/aws/2009/09/bioinformatics-genomes-ec2-and-hadoop.html.

[43] Amazon Simple Storage Service (Amazon S3). Acessado em 10 de Novembro de 2010., . URL http://aws.amazon.com/s3/.

[44] Openlink virtuoso. Acessado em 10 de Dezembro de 2010., . URL http://virtuoso.openlinksw.com/.

[45] SPARQL endpoint. Acessado em 02 de Fevereiro de 2011. URL http://semanticweb.org/wiki/SPARQL_endpoint.

[46] Virtuoso Open-Source Edition Functionality. Disponível em URL: http://virtuoso.openlinksw.com/dataspace/dav/wiki/Main/#Functionality%20Realms. Acessado em 10 de Janeiro de 2011.

[47] Large Triple Storages. Acessado em 10 de Fevereiro de 2011., . URL http://www.w3.org/wiki/LargeTripleStores.

[48] G. P. Telles, M. M. Brígido, N. F. Almeida, C. J. M. Viana, D. A. S. Anjos, and M. E. M. T. Walter. A Method for Comparing Three Genomes. *Lecture Notes in Computer Science*, 3594:160–169, 2005.

[49] Mogrovejo R.M., Viana, C.J.M., Tristão C., Bezerra M.M., and Lifschitz S. A Cloud-based Method For Comparing Three Genomes. poster proceedings ISSN-2178-5120, August 2010.

[50] The UniProt Consortium. The Universal Protein Resource (UniProt) in 2010. *Nucleic Acids Research*, 38(suppl 1):D142–D148, 2010. doi: 10.1093/nar/gkp846. URL http://nar.oxfordjournals.org/content/38/suppl_1/D142.abstract.

[51] Bio2RDF - Semantic web atlas of postgenomic knowledge. Acessado em 19 de Fevereiro de 2011. URL http://bio2rdf.org/.

[52] Hadoop 0.20 Documentation. Acessado em 02 de Março de 2011. URL http://hadoop.apache.org/common/docs/r0.20.2/index.html.

[53] Amazon Elastic Compute Cloud (Amazon EC2) - Pricing. Acessado em 10 de Novembro de 2010., . URL http://aws.amazon.com/ec2/#pricing.

[54] Amazon Simple Storage Service (Amazon S3) - Pricing. Acessado em 02 de Fevereiro de 2011, . URL http://aws.amazon.com/s3/#pricing.

[55] S3hub - amazon s3 client for mac os x. Acessado em 20 de Fevereiro de 2011. URL http://s3hub.com/.

[56] 35+ Amazon S3 Tools and Plugins for Windows, Mac and Linux. Acessado em 02 de Março de 2011., . URL http://www.zimbio.com/Amazon+EC2+and+S3/articles/89/35+Amazon+S3+Tools+Plugins+Windows+Mac+Linux.

[57] Best tools to manage Amazon S3 data. Acessado em 17 de Fevereiro de 2011, . URL http://indianblogger.com/best-tools-to-manage-amazon-s3-data/.

[58] S3 tools - command line S3 client. Acessado em 23 de Fevereiro de 2011. URL http://s3tools.org/s3cmd.

[59] Amazon Developer Tools. Acessado em 11 de Fevereiro de 2011, . URL http://aws.amazon.com/developertools?_encoding=UTF8&jiveRedirect=1.

[60] Amazon Elastic MapReduce Ruby Client. Acessado em 02 de Março de 2011. URL http://aws.amazon.com/developertools/2264?_encoding=UTF8&queryArg=searchQuery&binValue_2=Amazon%20Elastic%20MapReduce&fromSearch=1&binField_1=type&searchQuery=&binField_2=amzn-products&binValue_1=developertools.

[61] R2RML: RDB to RDF Mapping Language. Acessado em 10 de Novembro de 2010., . URL http://www.w3.org/TR/2011/WD-r2rml-20110324/.