Análise de Desempenho de Banco de Dados Relacionais e Não Relacionais em Dados Genômicos

Performance Analysis of Relational and Non Relational Databases on Genomic Data

Juccelino Rodrigues Alves de Barros [1, 2]
Gustavo Rau de Almeida Callou [1, 3]
Glauco Estácio Gonçalves [1, 4]
Victor Wanderley Costa de Medeiros [1, 5]
Carlos Henrique M. Castelletti [6, 7]

Data de submissão: 10/03/2017 , Data de aceite: 08/06/2017

Resumo: O armazenamento de dados genômicos é um grande desafio hoje, pois com o avanço da tecnologia molecular a quantidade de dados genômicos gerados está aumentando, de forma que o sequenciamento de um único organismo pode gerar arquivos com *terabytes* de informações. De forma geral, os processos de manipulação de dados genômicos fazem uso de simples arquivos como o principal meio para armazenamento de tais dados. Contudo, os bancos de dados modernos se apresentam como alternativa para a gerência desses dados por oferecer melhor organização, tolerância a falhas, melhor uso do espaço disponível para armazenamento e desempenho. Além disso, os bancos de dados permitem agregar aos dados brutos do sequenciamento meta-informações acerca das sequências de DNA armazenadas. Diante deste cenário, este trabalho apresenta e avalia o desempenho de diferentes estratégias de armazenamento em três bancos de dados pertencentes a dois paradigmas diferentes, o MySQL (representante dos bancos de dados Relacionais), o Cassandra e o MongoDB (representantes dos bancos de dados Não Relacionais). Os resultados demonstraram que os bancos de dados relacionais apresentam limitações quando estão inseridos em um ambiente com grandes massas de dados.

[1] Departamento de Estatística e Informática, UFRPE - Universidade Federal Rural de Pernambuco - Recife, Pernambuco, Brasil
[2] {juccelino.abarros@ufrpe.br}
[3] {gustavo.callou@ufrpe.br}
[4] {glauco.goncalves@ufrpe.br}
[5] {victor.wanderley@ufrpe.br}
[6] Laboratório de Imunopatologia Keizo Asami (LIKA), UFPE - Universidade Federal de Pernambuco - Recife, Pernambuco, Brasil
[7] {hcastelletti@prospecmol.org}

Abstract: Nowadays, advancements in molecular technology brought better equipments to obtain more DNA sequences in less time, making storage of genomic data a great challenge. As a result, the amount of genomic data generated has been increasing in a way that sequencing a single organism can generate terabytes of information. In general, genomic data processing make use of simple files as the primary means for storing these data. However, modern databases are a good alternative for improved management of these data by offering better organization, fault tolerance, use of available memory for storage, and performance, as they are made in order to optimize these tasks. Additionally, databases allow adding meta-information about the stored DNA sequences. Considering such a scenario, this paper presents and evaluates the performance of different storage strategies in three databases that belong to two different paradigms, MySQL (which belongs to the Relational Database paradigm), Cassandra, and MongoDB (which belong to Non-Relational Database paradigm). The results show that relational databases have limitations to cope with large genomic data sets.

***P*alavras-chave:** Dados genômicos, banco de dados, banco de dados relacional, banco de dados não relacional.

***K*eywords:** Genomic data, database, relational database, non-relational database.

1 Introdução

Os dados genômicos são gerados através do sequenciamento de algum material genético. Normalmente, os valores encontrados durante o sequenciamento de DNA são as 4 bases nitrogenadas: Adenina (A), Guanina (G), Citosina (C), Timina (T). Com o avanço da tecnologia molecular, esses dados são cada vez maiores, gerados mais rápido e com um custo menor. Os primeiros sequenciadores genéticos demoravam em torno de 45 dias para ler um gene (segmento de uma molécula de DNA). Hoje, com os atuais sequenciadores é possível sequenciar e analisar um genoma humano inteiro em 26 horas [1]. O custo para sequenciar um milhão de bases nitrogenadas vem reduzindo nos últimos anos. Partindo de 8 mil dólares em 2001 para menos de 10 centavos em 2015 [2].

Entre os bancos de dados genômicos mais famosos, está o *GenBank*[8], que é um banco de dados público que possui integração com outros bancos de dados genômicos distribuídos pelo mundo, entre eles o *Whole-Genome Shotgun* (WGS) [3]. Através do *GenBank* pode-se realizar comparações entre sequências de DNA identificando as espécies e a correlação entre elas. O gráfico representado na Figura 1 exibe o crescimento do número de sequências de DNA encontradas no *GenBank* e no *WGS*, este último sendo iniciado em meados de 2002. Observa-se que a quantidade de sequências geradas tende aumentar 10 vezes em intervalos de 5 anos. Isso quer dizer que, o avanço da tecnologia molecular está trazendo maiores arquivos com baixo custo e consequentemente novos projetos genomas estão surgindo.

São gerados, ao final do sequenciamento, arquivos com várias informações, dentre as quais se encontram as cadeias de DNA correspondentes ao material genético e as informações do sequenciador que realizou o procedimento. Esses arquivos tendem a ser grandes, alguns contendo *gigabytes* de informações, e como consequência disso, faz-se necessário um número elevado de recursos computacionais e de armazenamento para processar e guardar essas informações. O alto custo de armazenamento acaba sendo um grande problema para vários usuários que trabalham nessa área, pois alguns não dominam as melhores tecnologias para gerenciar esses dados.

[8] http://www.ncbi.nlm.nih.gov/genbank/

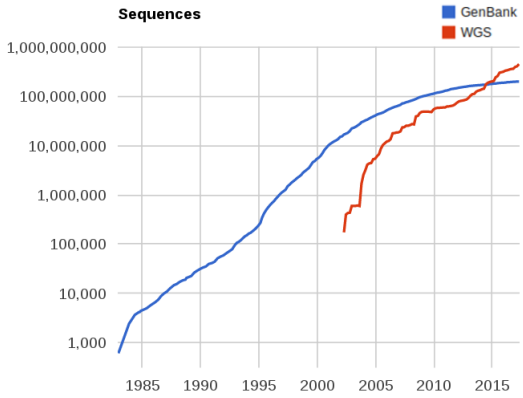

Figura 1. Número de sequências geradas no GenBank e WGS.

Atualmente, são encontradas algumas estratégias para o armazenamento dos dados genômicos, dentre elas, destaca-se a estratégia baseada em arquivos. Esses arquivos, em formato texto ou binário, são armazenados em conjunto em um repositório de arquivos. Esta abordagem traz muitas desvantagens, visto que as informações desses arquivos ficam dispersas no repositório, dificultando a recuperação de informações específicas.

Outra estratégia de armazenamento é a utilização de um banco de dados. A principal vantagem de um banco de dados, em vez de um repositório de arquivos, é a possibilidade de realizar consultas a informações específicas de um arquivo. Além disso, os bancos de dados oferecem regras de acesso e segurança para os usuários. Contudo, um banco de dados mal projetado ou que utilize estratégias inadequadas à gerência de dados genômicos pode introduzir mais malefícios do que benefícios ao usuário.

Encontrar a melhor forma de armazenar os dados genômicos é um grande desafio, devido ao alto custo de armazenamento e o conhecimento das melhores tecnologias para atender a essa demanda. O objetivo deste artigo é propor uma estratégia de armazenamento de dados genômicos em banco de dados relacionais, representado pelo MySQL, e banco de dados não relacionais, representados pelo Cassandra e MongoDB. Dessa forma, este artigo cria uma estratégia de armazenamento do conteúdo do arquivo genômico através da inserção de registro único e agrupados por linha de cada tabela no MySQL, MongoDB e Cassandra. Uma ferramenta foi desenvolvida com o objetivo de automatizar os experimentos e comparar o desempenho entre esses bancos, em termos de operações de inserção, consulta e extração.

O artigo está dividido conforme descrito a seguir. A Seção 2 mostra os trabalhos relacionados. A Seção 3 apresenta os dados genômicos e mostra alguns conceitos dos bancos de dados relacionais e não relacionais. Em seguida, a Seção 4 apresenta a metodologia utilizada nessa pesquisa. A Seção 5 ilustra a abordagem proposta através de um estudo de caso com análise de desempenho entre os bancos de dados escolhidos. Por último, a conclusão do artigo e trabalhos futuros são apresentados.

2 Trabalhos Relacionados

Alguns trabalhos discutem qual a melhor forma de armazenamento dos dados genômicos. Em [4] foi proposto o desenvolvimento de um modelo de dados para um *pipeline* de sequenciamento de alto desempenho, utilizando o MySQL. No trabalho, o próprio arquivo era armazenado em tabelas com campos do tipo BLOB (*Binary Large Object*). Além disso, o modelo de dados proposto tinha informações da

origem do dado genômico, do sequenciador, da qualidade do sequenciamento e da espécie sequenciada. Já em [5], foi utilizado o Cassandra para armazenar os mesmos arquivos encontrados em [4] para fins de comparação, desta vez era armazenado o conteúdo do arquivo em tabelas com várias colunas em que cada coluna tinha o conteúdo agrupado. O foco deste trabalho era realizar uma comparação entre o armazenamento do próprio arquivo (BLOB), proposto em [4], e o armazenamento do conteúdo do arquivo.

Röhm et al. [6] propõe a criação de um modelo de dados híbrido com as informações dos metadados dos arquivos e o próprio arquivo, armazenado em BLOB, utilizando o SQL Server 2008. Langille et al. [7] propõe o MicrobeDB, um projeto de código aberto[9] que oferece uma integração com o *National Center for Biotechnology Information* (NCBI[10]), um dos principais repositórios de dados genômicos, como também organiza e armazena os dados utilizando o MySQL. Este projeto tem o objetivo de facilitar o acesso aos dados, que são atualizados do NCBI para a banco de dados do projeto através da execução de *scripts* que podem ser programáveis para ser executado semanalmente, por exemplo.

Neste trabalho, identificaram-se quatro aspectos que devem ser considerados no projeto de um banco de dados genômico: número de colunas da tabela de armazenamento do arquivo; armazenamento das SRSs em estruturas BLOB ou em vários registros; tamanho do conjunto de SRSs por registro; e criação de uma tabela única ou de várias tabelas por arquivo. A Tabela 1 ilustra as estratégias que foram utilizadas nos trabalhos [4], [5] e [6] ordenados cronologicamente.

Tabela 1. Estratégias utilizadas nos trabalhos encontrados.

Estratégia	**[6] 2009, SQL Server**	**[4] 2012, MySQL**	**[5] 2014, Cassandra**	**Este trabalho**
Nº de colunas por tabela	1	1	10	1
Armazenamento SRS	BLOB	BLOB	registros	registros
Conj. de SRS por registro	-	-	5 (50 por linha da tabela)	1,5 e 50
Nº de tabelas por arquivo	1 para todos arquivos	1 para todos arquivos	1 por arquivo	1 por aquivo

Analisando a Tabela 1, é possível observar que os trabalhos encontrados não abordaram a estratégia de armazenar os registros agrupados em uma única coluna da tabela. Além disso, a operação de consulta por SRS foi pouco analisada nos trabalhos encontrados. Logo, definir um número fixo de SRSs armazenada por linha da tabela é uma abordagem pouco analisada em outros trabalhos. Nesta abordagem é possível observar o comportamento dos bancos de dados quando estão sendo armazenadas grandes cadeias de caracteres por campo da tabela, assim pode-se analisar se o tempo de inserção e de extração sofrem alguma influência. O armazenamento de uma única SRS por coluna da tabela tem como consequência um número maior de registros inseridos no banco de dados, mas é a única forma de realizar consulta por identificadores de sequências específicos, visto que quando inseridos em conjunto, os valores dos identificadores de sequências são concatenados de acordo com o tamanho definido de cada conjunto. Por outro lado, a vantagem de armazenar um conjunto de múltiplas SRSs por coluna da tabela é que o número de registros inseridos será menor quanto maior for esse conjunto.

Logo, diferente dos trabalhos encontrados, este artigo propõe uma estratégia de armazenamento do conteúdo dos dados genômicos através da inserção de registro único e agrupados por linha de cada

[9]https://github.com/mlangill/MicrobeDB
[10]http://www.ncbi.nlm.nih.gov/

tabela no MySQL, MongoDB e Cassandra. Além disso, foi realizada uma análise de desempenho entre esses bancos de dados em operações de inserção, consulta e extração.

3 Referencial Teórico

Esta seção descreve dois formatos de dados genômicos e mostra alguns conceitos dos bancos de dados relacional e não relacional.

3.1 Formato dos Dados Genômicos

Existem vários formatos de arquivos para dados genômicos, que podem ser encontrados em bancos de dados online como o *NCBI*. Entre os formatos mais comuns estão o *fasta* e o *csfasta*.

O formato *fasta*[8] é a forma mais simples de representar um sequenciamento, sendo dividido em duas partes: identificador de sequência, que é representado pelo sinal de “maior que", e a própria sequência de DNA, que é disposta na próxima linha após o seu identificador. A Figura 2 ilustra um exemplo de um arquivo *fasta*, primeira e terceira linha são os identificadores de sequência das cadeias de DNA presentes na segunda e quarta linha, respectivamente. Normalmente, esses arquivos são agrupados em *Short Read Sequence* (SRS), que é representado como uma tupla formada pelo identificador de sequência e a sequência de DNA referente a este identificador. Na Figura 2 podemos identificar duas SRSs.

```
>746_81_147_F3
CCACTACGTTCCACGGCACACTACTAAGTCCGTGTGACACACACACACAC
>746_81_203_F3
CCACTACGTTCCACGGCACACTACTAAGTCCGTGTGACACAGTGTGTGTG
```

Figura 2. Exemplo de um arquivo *fasta* com duas SRSs.

O *csfasta* é um tipo de arquivo proprietário [11] gerado pelo próprio sequenciador logo após o término do sequenciamento. Além de informações complementares no cabeçalho do arquivo, o formato *csfasta* armazena o identificador de sequência e a cadeia de DNA referente, que representa os nucleotídeos através de números. A Figura 3 ilustra um arquivo *csfasta*.

```
>sequence
T3122013131
```

Figura 3. Exemplo de um arquivo *csfasta*.

Esses números se referem às cores que o sequenciador gera quando está processando o material genético. Cada um desses números se referem a um possível par de DNA [9], como mostra a Tabela 2.

É possível realizar a conversão do arquivo *csfasta* para *fasta*. Para isso, é necessário conhecer as possíveis combinações de cada número. Considerando os números da Figura 3, o primeiro passo é realizar a combinação do nucleotídeo “T", que é o nucleotídeo padrão para iniciar o sequenciamento, com o número 3. Observando a Tabela 2 é constatado que essa combinação resulta no nucleotídeo “A". Este procedimento é feito a cada novo nucleotídeo encontrado. A tradução da sequência da Figura 3 para o arquivo *fasta* está na Figura 4.

[11] http://www.thermofisher.com/us/en/home.html

Tabela 2. Combinação do número com par de DNA. [9]

AA	CC	GG	TT	0	azul
AC	CA	GT	TG	1	verde
AG	CT	GA	TC	2	amarelo
AT	CG	GC	TA	3	vermelho

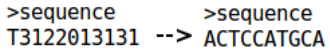

Figura 4. Conversão do arquivo *csfasta* para *fasta*.

3.2 Banco de Dados Relacional

Os bancos de dados são coleções de dados relacionados que seguem uma determinada estrutura, de forma que possam ser recuperados quando necessário [10]. Eles podem ser divididos em dois paradigmas: Relacional e Não Relacional. O banco de dados Relacional pode ser definido como uma forma de distribuição de dados de modo que eles tenham alguma relação representada por meio de tabelas [10]. Para garantir o funcionamento correto, eles seguem um conjunto de regras chamadas de formas normais. As três primeiras formas normais são fundamentais para garantir um mínimo de redundância para um bom modelo de dados [10]. Para executar as transações nesses bancos, deve-se respeitar as propriedades ACID (Atomicidade, Consistência, Isolamento, Durabilidade), com o objetivo de garantir o funcionamento correto do sistema e a respectiva consistência dos dados [11]. O MySQL[12] e o PostgreSQL[13] são exemplos de banco de dados deste paradigma.

O MySQL é um dos principais banco de dados relacionais. Sua arquitetura é composta por três camadas: conexão com o cliente, análise da instrução SQL e execução da instrução [12]. A primeira camada é onde ocorre a conexão das aplicações com o banco de dados. A segunda é responsável por otimizar a instrução, quando necessário. É nesta camada que são construídas as *procedures, triggers* e *views*. O MySQL possui um artefato chamado *Query Cache*, onde são armazenados os últimos comandos SQL para fins de otimização. A última camada é responsável por executar a instrução recebida da camada acima. Estas instruções são executadas no *Storage Engine*, o qual é responsável por armazenar e recuperar todos os dados.

3.3 Banco de Dados Não Relacional

Os bancos de dados não relacionais não apresentam todas as características ACID. Esses bancos se baseiam no teorema CAP (*Consistency, Availability* e *Partition tolerence*) que diz que em um dado momento, só é possível garantir duas das três propriedades entre a consistência, disponibilidade e a tolerância à partição [11]. Os bancos de dados orientado a objetos e o NoSQL (*Not Only SQL*) são exemplos de bancos que se encaixam nesse contexto [5].

Os bancos de dados NoSQL não utilizam a linguagem SQL para realizar suas transações. Eles apresentam um conjunto de conceitos que permitem o processamento de dados de forma rápida e eficiente com o foco em desempenho [13]. Representam uma alternativa para modelar dados sem se preocupar com

[12]https://www.mysql.com/
[13]https://www.postgresql.org/

os padrões rígidos propostos pelo modelo relacional. O NoSQL tem uma estrutura distribuída e tolerante a falhas que se baseia na redundância de dados em vários servidores.

Os modelos de banco de dados NoSQL são chave-valor, orientado a coluna, orientado a documento e orientado a grafos [14]. Cada modelo possui vantagens e desvantagens. Para manipulação de dados estatísticos, frequentemente escritos mas raramente lidos, pode ser usado um banco de dados do tipo chave e valor ou orientado a documentos [11]. Caso seja necessária uma aplicação com alta disponibilidade, na qual a minimização da inatividade é fundamental, é recomendado utilizar um banco de dados orientado a coluna. Para altos desempenhos de consultas complexas, recomenda-se o modelo orientado a grafos[11].

O Cassandra é um banco de dados orientado a coluna, criado para armazenar grandes quantidades de dados espalhados por vários servidores e, mesmo assim, oferecer alta viabilidade de acesso a dados consistentes [15]. Este banco de dados foi baseado na arquitetura do Dynamo da Amazon[14] e o no modelo de dados do Bigtable da Google[15]. O foco principal do Cassandra é no desempenho das consultas. Possui uma linguagem própria chamada CQL (*Cassandra Query Language*), semelhante ao SQL. É um banco de dados distribuído, descentralizado, altamente disponível e escalável horizontalmente. Os dados são distribuídos entre várias instâncias desse mesmo banco de dados que estejam executando em outras máquinas.

O MongoDB é um banco de dados orientado a documentos e de código livre. Surgiu em 2009, foi implementado em C++ [16]. Documentos no MongoDB são objetos JSON que possuem um conjunto de campos. O campo é uma chave do objeto que foi inserido, enquanto que o valor pode ser um texto, outro documento, um inteiro, um ponto flutuante, um binário, uma data entre outros. Este banco de dados possui uma linguagem para realizar as transações completamente diferente do CQL e do SQL. É uma linguagem mais simples, necessitando de poucas linhas para executar uma instrução. O MongoDB foi projetado para ser livre de esquema, ou seja, o usuário não se preocupa em como os dados têm que ser inseridos.

4 Metodologia

Esta seção apresenta o processamento dos dados genômicos e a estratégia utilizada para o armazenamento desses dados.

4.1 Processamento dos Dados Genômicos

Após realizado o sequenciamento do material genético, o sequenciador gera um conjunto de arquivos texto que indicam a qualidade, as informações sobre o sequenciador e a própria sequência de DNA. O arquivo que possui todo o conteúdo, na maioria das vezes, se encontra no formato *csfasta*. Após esse procedimento, a sequência encontrada pode ser tratada e utilizada das mais variadas formas, de acordo com o estudo em questão.

Depois do sequenciamento, os arquivos de dados genômicos são geralmente armazenados em repositório de arquivos. Esta estratégia leva ao problema de localizar as informações a respeito desses arquivos, pois os seus meta-dados não são armazenados ou são difíceis de localizar [6]. Assim, o uso de um banco de dados evitar essas desvantagens, comparado ao armazenamento em repositório de arquivos.

[14]http://aws.amazon.com/pt/documentation/dynamodb/
[15]https://cloud.google.com/bigtable/

A Figura 5 apresenta como um banco de dados pode ser integrado à sequência de passos adotados para se realizar o processamento dos dados genômicos. As etapas envolvem desde a filtragem dos arquivos *csfasta* até o processamento dos arquivos *fasta* utilizados por meio de ferramentas de análise das sequências de DNA. Este trabalho tem foco na parte destacada da figura, na qual as etapas referentes tanto a inserção como a extração dos dados no banco são realizadas.

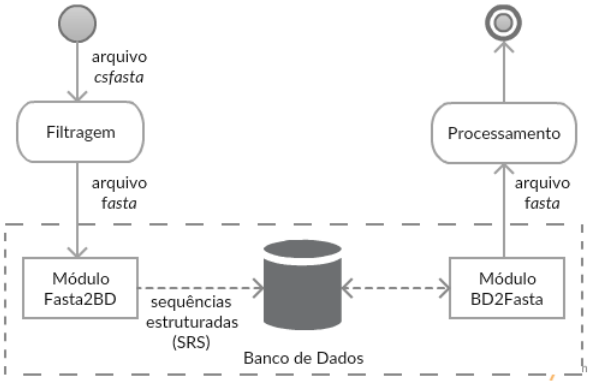

Figura 5. Integração de um banco de dados no processo de tratamento dos arquivos *fasta*.

O objetivo da etapa de filtragem é remover as sequências de baixa qualidade e indesejadas do arquivo *csfasta*, e gerar como resultado o arquivo *fasta*. A filtragem do arquivo *csfasta* para o *fasta* é feita utilizando *softwares* de tradução. Após a criação desse arquivo, são criadas as sequências estruturadas - *Short Read Sequences* (SRS) e inseridas no banco de dados através do módulo Fasta2BD. Já o módulo BD2Fasta realiza extrações de todo conteúdo e também pode realizar consultas por uma SRS, de modo a localizar sequências específicas de DNA. Os dados obtidos na extração do conteúdo são armazenados novamente no formato *fasta*. O armazenamento dos dados extraídos do banco de dados em arquivos deste tipo é essencial, já que as ferramentas de análise das sequências de DNA, utilizadas na fase de processamento, comumente reconhecem apenas os arquivos desse formato.

5 Estudo de Caso

Esta seção mostra um experimento realizado com dados genômicos em 3 bancos de dados: MySQL, representando o banco de dados relacional, MongoDB e Cassandra, representando o banco de dados não relacional. O objetivo do experimento é fazer uma análise de desempenho comparativa entre os bancos de dados relacional e não relacional em operações típicas como inserção, extração e consulta para se definir qual é o mais indicado para armazenamento dos dados genômicos.

5.1 Ambiente

O ambiente de teste organizado é composto por três computadores conectados em uma rede local *Gigabit Ethernet*. Esses computadores possuem 8GB de memória RAM, processador Intel i5, 1TB de espaço em disco e sistema operacional Ubuntu 14.04 LTS. Os bancos de dados MongoDB e MySQL foram instalados em uma única máquina. Já o Cassandra, por ser um banco de dados distribuído, foi instalado em 2 máquinas, porém foram removidos 4 GBs de memória RAM de cada máquina, tentando equilibrar os recursos computacionais.

Os experimentos foram todos automatizados por meio de uma ferramenta de benchmark, chamada de FastaBD[16], desenvolvida ao longo deste trabalho. Esta ferramenta permitiu medir os tempos individuais de inserção, extração e consulta aos bancos a partir de diferentes arquivos de dados genômicos. Ela realiza a função dos módulos Fasta2BD e BD2Fasta, exibido na Figura 5. A ferramenta foi executada em um computador exclusivo. A Figura 6 ilustra o ambiente de teste montado.

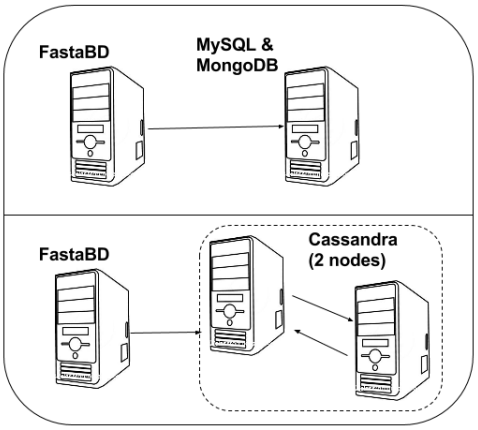

Figura 6. Ambiente de Teste.

Os arquivos *fasta* foram obtidos junto ao Laboratório de Imunopatologia Keizo Asami (LIKA)[17] a partir da filtragem de arquivos *csfasta* gerados no sequenciamento do material genético de caprinos. Foram escolhidos 4 arquivos de tamanhos variados com o objetivo de avaliar o comportamento do banco de dados nesses cenários distintos. Sendo assim, foram utilizados quatro arquivos *fasta*, chegando ao total de aproximadamente 2,3 *gigabytes* e 69.848.142 linhas. A Tabela 3 mostra detalhes de cada arquivo *fasta*.

Tabela 3. Características dos arquivos *fasta* utilizados neste trabalho.

Arquivo	**Local da Amostra**	**Tamanho (MB)**	**Nº de SRS**
cabra1.fa	próstata	763	11.756.047
cabra2.fa	glândula mamária	238	3.802.481
cabra3.fa	ovário	31	473.886
cabra4.fa	hipófise	1331,2	18.891.657
-	-	total aprox. 2,3 GB	34.924.071

Cada cadeia de DNA nos arquivos *fasta* deste estudo possui em torno de 50 caracteres, os identificadores de sequência não podem ser repetidos no mesmo arquivo, mas podem existir identificadores iguais em arquivos diferentes. No final da inserção dos arquivos com o número de SRS por linha da tabela igual 1, por exemplo, eram inseridos 34.924.071 registros no banco de dados. Para definir a quantidade de registros que são inseridos no banco de dados, divide-se o total de SRS do arquivo pela quantidade de SRS inserida por registro.

[16]https://github.com/jukabarros/fastaExperiments
[17]https://www.ufpe.br/lika/

5.2 Esquemas dos Bancos de Dados

Os esquemas dos bancos de dados foram projetados baseados nas características dos paradigmas Relacional e Não Relacional. O esquema do MySQL tem duas tabelas: *fasta_info* e *fasta_collect*. A primeira tabela armazena informações dos arquivos existentes no banco de dados, como nome, número de linhas e tamanho. Já a tabela *fasta_collect* armazena o conteúdo do arquivo. Existem 3 campos nesta tabela: identificador de sequência, cadeia de DNA e a linha referente à localização do identificador de sequência. Os demais campos desta tabela são a chave primária, que é um valor incrementado automaticamente, e a chave estrangeira referente à tabela *fasta_info*. A relação da tabela *fasta_info* é de 1 para N com a tabela *fasta_collect*. A Figura 7 ilustra o esquema do MySQL.

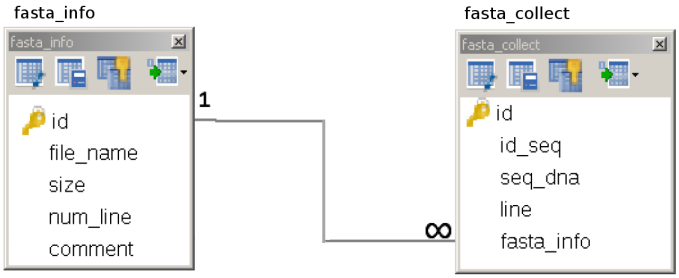

Figura 7. Esquema do MySQL.

Enquanto o MySQL tem um esquema bem definido e estático, o MongoDB e o Cassandra possuem um esquema totalmente dinâmico. A estratégia de usar esquema dinâmico é justificada pelo fato de que os dois bancos de dados NoSQL têm como característica uma modelagem livre de esquema. Os esquemas desses bancos de dados são representados da seguinte maneira: uma tabela com a mesma estrutura *fasta_info* e a cada arquivo inserido é criada uma nova tabela com o mesmo nome desse arquivo, que possui os mesmos campos da tabela *fasta_collect* com a exceção da chave estrangeira, que não existe, e da chave primária, que passa a ser o identificador de sequência. A Figura 8 ilustra o esquema do Cassandra e do MongoDB.

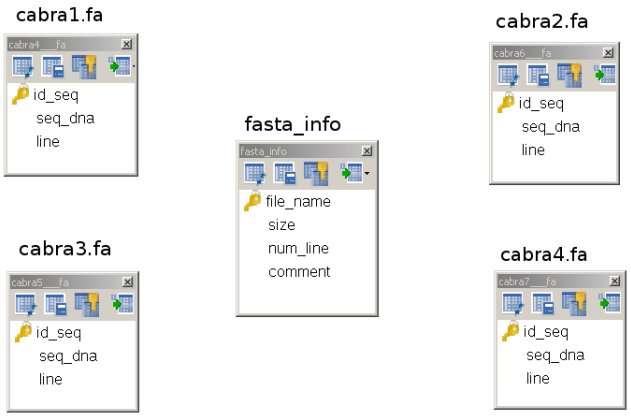

Figura 8. Esquema do Cassandra e do MongoDB.

5.3 Configuração do Experimento

Cada experimento nesta avaliação consiste em três etapas executadas nesta ordem: inserção, extração e consulta. Na inserção, as SRSs são lidas do arquivo sequencialmente e tratadas de acordo com o parâmetro do número de SRSs por linha para ser enviada ao banco de dados. Os arquivos foram inseridos na ordem que estavam distribuídos no diretório e o tempo de inserção individual de cada arquivo foi registrado. A extração consiste no processo inverso da inserção, onde são obtidas todas as SRSs referentes a cada um dos arquivos para sua remontagem. Por fim, na etapa da consulta, realiza-se a recuperação de 30 SRSs diferentes a partir de seu identificador e o tempo desta consulta da cada SRS é medido.

Dois fatores foram utilizados neste experimento: número de SRSs por linha e o banco de dados escolhido. Para o número de SRSs por linha foram utlizados três níveis: 1, 5 e 50 SRSs por linha da tabela, enquanto que os bancos de dados escolhidos foram o MySQL, Cassandra e MongoDB. Apesar de pertencerem ao mesmo paradigma, o Cassandra e o MongoDB, ambos NoSQL, apresentam abordagens diferentes para o gerenciamento de dados. Como visto, o Cassandra é orientado a colunas, enquanto o MongoDB é orientado a documentos. A ideia é analisar o comportamento destes 2 bancos de dados, incluindo também o MySQL, que apresenta uma abordagem relacional de dados, em um ambiente com dados genômicos. Para cada combinação dos níveis dos fatores escolhidos (9 combinações) foram obtidas amostras suficientes a fim de obter resultados estatisticamente significativos e permitir conclusões confiáveis. Para atingir esses resultados, foram coletadas 30 amostras.

Para os níveis de SRSs por linha 5 e 50, a etapa de consulta não foi realizada, visto que o armazenamento de múltiplas SRSs em um único registro impossibilita a realização da consulta por somente um identificador de SRS, já que elas estão concatenadas de acordo com o tamanho do conjunto. Por isso, a etapa de consulta foi realizada somente quando o número de SRS por linha na tabela era um (1). A Figura 9 ilustra o diagrama de atividades do experimento.

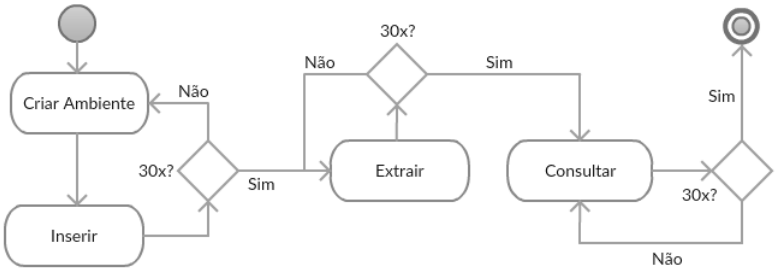

Figura 9. Diagrama de Atividades do Experimento.

5.4 Resultados

Os resultados obtidos estão separados de acordo com cada etapa do experimento. Os tempos encontrados nos gráficos e nas tabelas estão em segundos e os tamanhos dos arquivos foram mensurados na ordem de *megabytes*. O intervalo de confiança para a média, considerando 95% de confiança, foi calculado e, na maioria dos casos, os intervalos não se sobrepõem, mostrando que os valores das médias são estatisticamente diferentes. Por isso, os intervalos de confiança não serão mostrados nos gráficos, mas os casos em que não foi possível definir a diferença estatística serão pontuados no texto.

Inserção. A Figura 10 mostra o gráfico da média do tempo de inserção do arquivo cabra1.fa em cada um dos bancos de dados com a variação do número de SRSs por linha. As Tabelas 4, 5 e 6 mostram os valores

estatísticos dos demais arquivos para o número de SRS 1, 5 e 50 respectivamente. Pode-se observar que os tempos são proporcionalmente similares.

O MongoDB possui o menor tempo de inserção na maioria das situações. Enquanto o MySQL apresenta o maior tempo em todas as situações. Contudo, à medida que o número de SRS por linha da tabela aumenta, a diferença do tempo de inserção entre os três bancos de dados diminui. No caso em que este número é igual a 50 para o arquivo cabra2.fa, não é possível chegar a uma conclusão do melhor tempo entre o Cassandra e o MongoDB, pois eles são estatisticamente iguais (Tabela 6). É possível concluir que os bancos NoSQLs apresentaram bons resultados na inserção comparado ao MySQL. Quanto maior for a cadeia de caracteres, melhor será o desempenho dos 3 bancos de dados. No quesito estabilidade, o MongoDB apresentou melhores resultados, visto que o desvio padrão em todas as situações foi o menor, conforme as Tabelas 4, 5 e 6.

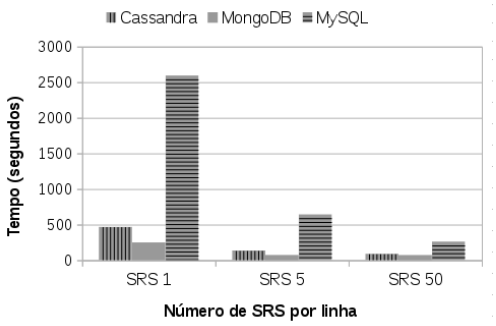

Figura 10. Tempo médio de Inserção - cabra1.fa (763MB).

Tabela 4. Valores Estatísticos do tempo (segundos) de inserção – SRS 1

cabra2.fa (238MB)			
Bancos de Dados	**Média**	**Desv. Padrão**	**Int. Confiança (95%)**
Cassandra	151,07	4,66	[149,40 : 152,74]
MongoDB	80,29	1,14	[79,88 : 80,70]
MySQL	844,48	6,04	[842,32 : 846,46]
cabra3.fa (31MB)			
Cassandra	18,45	2,37	[17,60 : 19,30]
MongoDB	10,38	0,35	[10,26 : 10,50]
MySQL	106,41	5,62	[85,37 : 86,97]
cabra4.fa (1,3GB)			
Cassandra	798,95	24,54	[790,17 : 807,73]
MongoDB	401,88	4,29	[400,34 : 403,42]
MySQL	4198,48	14,36	[4193,34 : 4203,62]

Tabela 5. Valores Estatísticos do tempo (segundos) de inserção – SRS 5

cabra2.fa (238MB)			
Bancos de Dados	**Média**	**Desv. Padrão**	**Int. Confiança (95%)**
Cassandra	48,36	4,62	[46,98 : 50,28]
MongoDB	22,98	1,18	[22,56 : 23,40]
MySQL	211,37	2,05	[210,64 : 212,10]
cabra3.fa (31MB)			
Cassandra	5,47	1,00	[5,11 : 5,83]
MongoDB	3,15	0,09	[3,12 : 3,18]
MySQL	26,80	1,71	[26,19 : 27,41]
cabra4.fa (1,3GB)			
Cassandra	260,64	15,84	[254,97 : 266,31]
MongoDB	117,73	1,29	[117,27 : 118,19]
MySQL	1058,19	5,90	[1056,08 : 1060,30]

Tabela 6. Valores Estatísticos do tempo (segundos) de inserção – SRS 50

cabra2.fa (238MB)			
Bancos de Dados	**Média**	**Desv. Padrão**	**Int. Confiança (95%)**
Cassandra	23,98	1,51	[21,44 : 24,52]
MongoDB	21,78	0,91	[21,45 : 22,11]
MySQL	86,17	2,24	[85,37 : 86,97]
cabra3.fa (31MB)			
Cassandra	3,35	0,28	[3,25 : 3,45]
MongoDB	3,00	0,19	[2,93 : 3,07]
MySQL	11,17	0,51	[10,99 : 11,35]
cabra4.fa (1,3GB)			
Cassandra	142,29	9,21	[138,99 : 145,59]
MongoDB	117,87	1,30	[117,40 : 118,34]
MySQL	441,16	4,56	[439,53 : 442,79]

Extração. As Figuras 11 e 12 mostram os gráficos da média do tempo de extração do menor arquivo (cabra3.fa) e do maior (cabra4.fa) em cada um dos bancos de dados com a variação do número de SRSs por linha da tabela.

No menor arquivo, o MongoDB é estatisticamente igual ao MySQL enquanto que o Cassandra apresenta o maior tempo, independente da variação da SRS. Analisando o eixo y do gráfico da Figura 11, é possível identificar que os tempos ficaram muito próximos e o Cassandra demorou alguns segundos a mais para realizar o experimento. Uma justificativa para esse comportamento é o tempo que o Cassandra leva para abrir e fechar a conexão com a aplicação cliente.

Já no maior arquivo (Figura 12), o MongoDB obteve os melhores tempos independente do número de SRSs. Enquanto o MySQL teve o maior tempo médio de extração quando o número de SRS era igual a 1. O tempo mais longo de extração do MySQL é justificado pelo fato das consultas serem paginadas para grandes números de registros, ou seja, quando o número de registros de um arquivo específico inseridos no banco de dados era maior que 500.000, as consultas eram divididas em blocos de 500.000, valor escolhido baseado no número de SRS do arquivo cabra3.fa. As extrações foram realizadas dessa forma, pois o MySQL consome muitos recursos computacionais para realizar essas consultas e retornar os resultados. A análise do consumo de memória RAM e CPU na extração não-paginada revelou que a CPU atingia o limite máximo de uso e a memória RAM atingia aproximadamente 50%, tornando impossível fazer a extração de uma vez só independente do número de registros inseridos.

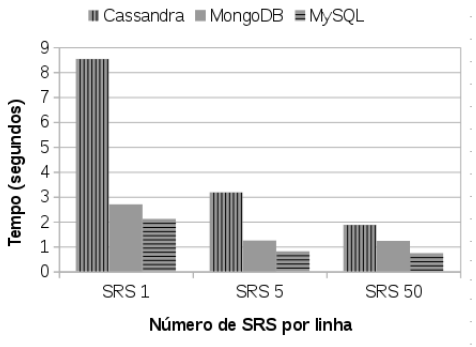

Figura 11. Média de Tempo de Extração - cabra3.fa (31MB).

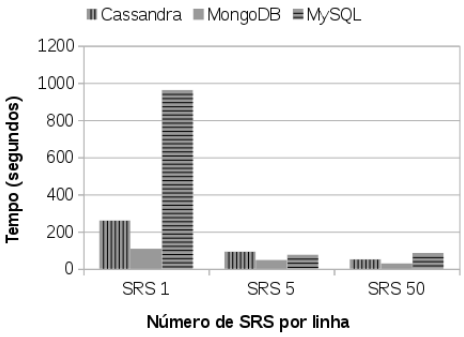

Figura 12. Média de Tempo de Extração - cabra4.fa (1,3GB).

Consulta. As consultas foram realizadas com 30 identificadores de sequências escolhidos aleatoriamente encontrado em qualquer arquivo armazenado. A Tabela 7 mostra os valores estatísticos do experimento. Lembrando que esse experimento foi realizado somente quando era inserido 1 SRS por linha da tabela, conforme explicado na seção 5.3.

Tabela 7. Valores estatísticos do tempo (segundos) da consulta.

Valores	Cassandra	MongoDB	MySQL
Média	0,49	23,43	30,07
Mediana	0,48	23,52	29,86
Desvio Padrão	0,04	0,58	0,97
Mínimo	0,45	22,02	28,58
Máximo	0,67	24,34	31,79
Int. de Conf. (95%)	[0,48 : 0,50]	[23,22 : 23,64]	[29,72 : 30,42]

A partir da Tabela 7, observa-se que o Cassandra possui o melhor tempo de consulta, sendo aproximadamente 48 vezes mais rápido que o MongoDB e 60 vezes mais rápido que o MySQL. O MySQL apresentou o pior tempo de consulta. Comparando o MongoDB com o MySQL, pode-se observar que o banco NoSQL possui o tempo médio de consulta e desvio padrão menor que o do banco relacional (23,43s vs 30,07s).

Discussão. Baseado nos resultados encontrados nos experimentos, observa-se que a estratégia de armazenar conjunto de SRSs por linha da tabela pode ser válida quando o usuário não precisa realizar consultas específicas. De modo geral, o MongoDB teve o melhor tempo de inserção na maioria das situações seguido pelo Cassandra. Quanto maior o número de SRS por linha da tabela, menor o tempo de inserção. Conclui-se que armazenar grandes cadeias de caracteres não prejudica o tempo de inserção dos dados.

Na extração em arquivos pequenos, o MySQL obteve bons resultados, porém na extração dos maiores arquivos era necessário realizar a paginação, devido ao alto consumo de recursos computacionais. Contudo, o MongoDB teve o melhor tempo de extração nas demais situações. Foi constatado também que armazenar grandes cadeias de caracteres não prejudica o tempo de extração dos dados. Já na consulta, o Cassandra obteve os melhores resultados, sendo até 60 vezes melhor que os outros bancos de dados. A Figura 13 mostra um quadro comparativo entre os três bancos de dados em cada etapa do experimento levando em consideração cenários com grande quantidade de dados. O MongoDB obteve a melhor qualificação nos quesitos inserção e extração, pois apresentou os melhores resultados na maioria das situações. Na inserção, o MongoDB obteve o melhor resultado em todos os cenários propostos. Para operação de consulta, destaca-se o Cassandra visto que seu tempo foi o menor entre os bancos de dados. O MongoDB e o MySQL apresentaram tempos maiores que o Cassandra, justificando a qualificação mínima.

Figura 13. Quadro comparativo entre os 3 bancos de dados.

6 Conclusão

Este trabalho abordou o problema de armazenar os dados genômicos, visto que eles estão em crescimento devido à modernização dos equipamentos que geram esses dados. O problema de armazenar os dados genômicos ainda não foi resolvido. Porém, os bancos de dados não relacionais podem ser uma boa alternativa para o armazenamento de dados genômicos, visto que, apresentam bom comportamento quando inserido em um ambiente com muitos dados. Esses resultados foram observados no estudo de caso realizado, em que o banco de dados relacional mostrou-se inadequado para o volume e o tipo de dados proposto.

Como trabalhos futuros, pretende-se avaliar outros bancos de dados relacionais e NoSQL. Assim como explorar ainda mais o aumento do número de SRS por linha da tabela.

7 Agradecimentos

Os autores gostariam de agradecer à FACEPE e ao CNPq pelo suporte financeiro a esta pesquisa.

8 Contribuição dos Autores

Os autores contribuíram igualmente na realização deste trabalho.

Referências

[1] MILLER, N. A. et al. A 26-hour system of highly sensitive whole genome sequencing for emergency management of genetic diseases. *Genome medicine*, BioMed Central, v. 7, n. 1, p. 100, 2015.

[2] Disponível em: <https://www.genome.gov/sequencingcostsdata/>. Acesso em: jul. 2016.

[3] LIPMAN, D. J.; OSTELL, J.; SAYERS, E. W. Dennis a. benson, mark cavanaugh, karen clark, ilene karsch-mizrachi. *Nucleic Acids Research*, Oxford Univ Press, v. 1, p. 7, 2012.

[4] HUACARPUMA, R. C. Modelo de dados para um pipeline de seqüenciamento de alto desempenho transcritômico. 2012.

[5] ANICETO, R. C.; XAVIER, R. F. Um estudo sobre a utilização do banco de dados nosql cassandra em dados biológicos. 2014.

[6] RÖHM, U.; BLAKELEY, J. Data management for high-throughput genomics. *arXiv preprint arXiv:0909.1764*, 2009.

[7] LANGILLE, M. G. et al. Microbedb: a locally maintainable database of microbial genomic sequences. *Bioinformatics*, Oxford University Press, v. 28, n. 14, p. 1947–1948, 2012.

[8] Disponível em: <http://blast.ncbi.nlm.nih.gov/blastcgihelp.shtml>. Acesso em: jul. 2016.

[9] Disponível em: <https://www.biostars.org/p/43855/>. Acesso em: mar. 2017.

[10] DATE, C. J. *Introdução a sistemas de bancos de dados*. São Paulo, Brasil: Elsevier Brasil, 2004.

[11] LÓSCIO, B. F.; OLIVEIRA, H. R. d.; PONTES, J. C. d. S. Nosql no desenvolvimento de aplicações web colaborativas. *VIII Simpósio Brasileiro de Sistemas Colaborativos*, v. 10, n. 1, p. 11, 2011.

[12] SCHWARTZ, B.; ZAITSEV, P.; TKACHENKO, V. *High performance MySQL: optimization, backups, and replication*. Sebastopol, USA: O'Reilly Media, Inc, 2012.

[13] MCCREARY, D.; KELLY, A. *Making sense of NoSQL*. Shelter Island, USA: Shelter Island: Manning, 2014. 19–20 p.

[14] CARNIEL, A. C. et al. Query processing over data warehouse using relational databases and nosql. In: *2012 XXXVIII Conferencia Latinoamericana en Informatica (CLEI)*. Medellin, Colombia: IEEE, 2012. p. 1–9.

[15] LAKSHMAN, A.; MALIK, P. Cassandra: a decentralized structured storage system. *ACM SIGOPS Operating Systems Review*, ACM, v. 44, n. 2, p. 35–40, 2010.

[16] Disponível em: <http://docs.mongodb.org/manual/>. Acesso em: mar. 2017.