Benchmarking do banco Cassandra utilizando dados do Exame Nacional do Ensino Médio (Enem)

Leonardo de Holanda Bonifácio[1], **Samuel Victor C. da Ponte**[1]

[1]Programa de Pós-Graduação em Computação Aplicada – Universidade de Brasília (UnB)
Brasília – Brazil

{leonardo.bonifacio, samuel.ponte}@aluno.unb.br

Abstract. *With the growing use of data in many areas of technology, it is very necessary to understand and study databases prepared to handle such volume of data. NoSQL databases have as main characteristic their scalability, being able to easily distribute the data among nodes and servers, becoming a viable alternative to the exposed problem. In order to evaluate Apache Cassandra's performance under high demands of read queries, this article performs a benchmarking experiment using data from the Exame Nacional do Ensino Médio (Enem), checking the influence of scalability factors, such as number of nodes in the cluster and the amount of data in the database, in the task performance.*

Resumo. *Com o crescimento de uso de dados em diversas áreas da tecnologia, se faz cada vez mais necessário o entendimento e estudo de bancos de dados preparados para lidar com tal volume. Os bancos NoSQL têm como característica principal a escalabilidade, sendo capazes de distribuir facilmente os dados entre um número de nós e servidores, se tornando uma alternativa viável ao problema exposto. Com o objetivo de avaliar o desempenho do Apache Cassandra sob demandas altas de consultas de leitura, esse artigo faz um experimento de benchmarking utilizando dados do Exame Nacional do Ensino Médio (Enem), verificando a influência de fatores de escalabilidade, como número de nós no cluster e quantidade de dados no banco, no desempenho da tarefa.*

1. Introdução

Bancos de dados *NoSQL* são aqueles projetados para conseguir lidar com volumes de dados considerados grandes (Big Data). Eles são pois, desde sua origem, estruturados de maneira a receber um fluxo intenso e contínuo de informações mantendo a qualidade de suas tratativas. Os modelos *NoSQL* também apresentam mais flexibilidade, não necessitando de descrição de dados e relacionamentos. Alguns, inclusive, não trazem definição de esquema [Silveira et al. 2020].

Dentre as diversas ferramentas disponíveis classificadas como *NoSQL*, existe o Apache Cassandra. Egresso do Facebook e posteriormente mantido pela comunidade de desenvolvedores, o banco consegue lidar de maneira satisfatória com grandes quantidades de informação. Nele, é possível adicionar nós (mais servidores) de maneira prática, e entre os fatores que o destacam como banco desta categoria, pode-se mencionar o baixo custo de manutenção, fácil escalabilidade, alta disponibilidade e ausência de ponto único de falha.

A fim de analisar sua performance, diversos estudos realizam *benchmarking* do Cassandra com ferramentas conhecidas, tais como o *Yahoo! Cloud Serving Benchmark* (YCSB). Nela, *workloads* pré-definidos de leitura, escrita e combinações são utilizados para testar a resposta do banco à variações de parâmetros, quantidade de dados, nós e *threads*.

O Exame Nacional do Ensino Médio (Enem) é uma forma anual de avaliar alunos recém saídos do Ensino médio e, também, uma maneira de ajudá-los a ingressar em instituições de ensino superior, particulares, públicas e também de fora do país. Os dados das provas e dos participantes são disponibilizados de forma gratuita na *internet* e os mesmos foram utilizados nesta pesquisa como registros de entrada do banco para testes.

Desta forma, com o intuito de investigar o impacto de fatores de escalabilidade no desempenho de execução de consultas de leitura, este artigo propõe, realiza e analisa um experimento de *benchmarking* do banco Cassandra utilizando dados do Enem. Esta investigação pode auxiliar na tomada de decisão para escolha do banco para uma possível aplicação e como fazer o ajuste/seleção dos fatores analisados. Com os resultados do experimento, as seguintes Questões de Pesquisa (QP) são respondidas:

- **QP1**: A quantidade de dados no banco influencia no desempenho de consultas de leitura?
- **QP2**: O aumento do número de nós favorece o desempenho da velocidade das consultas?

O restante do artigo é divido da seguinte forma: na Seção 2, são explicados os principais conceitos relacionados ao Cassandra e ao Enem. A Seção 3 apresenta trabalhos já realizados no mesmo campo de pesquisa, sobre o banco em questão e *benchmarking* em geral. Na Seção 4, a preparação e execução do experimento são detalhadas. A Seção 5 apresenta e discute os resultados obtidos e, por fim, na Seção 6 é feita uma breve conclusão do artigo, com propostas para trabalhos futuros.

2. Referencial Teórico

2.1. Cassandra

O Apache Cassandra é um banco de dados *NoSQL* que é classificado como parte do tipo Família-Coluna. Sua origem foi dada em 2007 no Facebook e hoje é mantida pela Apache Software. Esse banco foi desenvolvido com o objetivo de lidar com grandes quantidades de dados distribuídos e consultas, enquanto mantém boa consistência e distribuição de dados ao longo do *cluster* [Abramova et al. 2014].

Dentre as características que tornaram o Cassandra atrativo, pode-se citar as de que não tem ponto único de falha, possui baixo custo de propriedade e a possibilidade de escalar em sistemas distribuídos. Além disso, ter um *non-master cluster* é muito importante, visto que se um dos nós quebrar, isso não afeta a habilidade do *cluster* de continuar realizando suas tarefas de leitura e escrita no banco. Sendo assim, um *cluster* Cassandra é considerado de alta disponibilidade para leitura e escrita [Kozma and Morschheuser 2019].

Em [Mansouri and Babar 2020], o Cassandra é descrito como um banco *open-source* baseado nas ideias do Google BigTable e Amazon Dynamo. A possibilidade de ajustar em alto nível a consistência de acordo com os requisitos da aplicação é de grande

valor, realizando um balanço entre latência e consistência. Sua operação em modo *master-master* torna o escalonamento horizontal fácil de manter. Além disso, ele também provê diversas técnicas de particionamento e replicação.

2.2. Exame Nacional do Ensino Médio - Enem

O Enem foi instituído em 1998 no Brasil, e tem o objetivo de avaliar o desempenho escolar de estudantes ao término da educação básica. Em 2009, o exame deixou de ser apenas um teste e passou a ser utilizado como mecanismo de acesso à educação superior. As notas obtidas podem ser usadas para acesso ao Sistema de Seleção Unificada (Sisu) e ao Programa Universidade para Todos (ProUni). Com isso, os aplicantes podem ingressar em faculdades espalhadas pelo Brasil, serem aceitos em mais de 50 instituições de educação superior portuguesas e pleitear financiamento estudantil com o Fundo de Financiamento Estudantil (Fies), programa do governo.

Qualquer pessoa que concluiu o ensino médio ou está concluindo essa etapa pode fazer o Enem. Para o segundo caso, os participantes realizam o exame como ”treineiros”e o resultado serve apenas para autoavaliação de conhecimento.

Dentre as áreas de conhecimento, estão presentes: linguagens, códigos e suas tecnologias; ciências humanas e suas tecnologias; ciências da natureza e suas tecnologias; e matemática e suas tecnologias. As áreas acumulam 180 questões objetivas. Faz parte da avaliação, também, uma redação, que exige o desenvolvimento de um texto dissertativo-argumentativo através de uma situação problema [INEP 2021].

3. Trabalhos Relacionados

Os trabalhos relacionados à avaliação de performance do Cassandra se dividem em diversas subcategorias. Dentre as mais comuns, é possível citar: avaliação do número de *threads*, número de nós em um *cluster*, quantidade de dados no banco, influência da distância entre nós para nuvens híbridas, comparação com bancos da mesma categoria, dentre outros.

A maioria dos trabalhos utiliza como ferramenta de benchmarking o *Yahoo! Cloud Serving Benchmark* (YCSB), que permite a execução e medição de diversas operações em bancos de dados como HBase, Cassandra, MongoDB, Redis. As operações selecionam, através de uma distribuição, quais dados serão utilizados nos testes. As três distribuições disponíveis são: Uniforme (seleção aleatória), Zipfian (seleção de popularidade pela ferramenta), Última (dados inseridos por último têm maior chance) e Multinomial (probabilidade pode ser especificada pra cada item) [Kozma and Morschheuser 2019]. Com a ferramenta, é possível selecionar *workloads* pré-definidos de leitura, escrita, inserção, atualização e, também, a combinação dessas opções para obtenção de resultados completos.

Em [Mansouri and Babar 2020], é estudada a influência da distância entre nuvens híbridas na performance de diversos bancos como MongoDB, Cassandra, Riak, Redis. Quando há a combinação de nuvens públicas (que disponibilizam seus recursos de armazenamento, rede e computação para o público geral da *internet*) e nuvens privadas (recursos próprios de uma organização ou empresa), é necessário considerar a distância física entre os servidores. O artigo mostra como tal fator impacta na performance das operações no que diz respeito à latência, consistência e escalabilidade.

No artigo [Gorbenko et al. 2020], é analisada a performance do Cassandra com estudos relacionados ao *trade-off* entre consistência de dados e latência em sistemas de dados distribuídos. O banco citado disponibiliza uma série de parâmetros que possibilitam o ajuste de consistência e impactam diretamente outros fatores como latência de rede. Os autores explicitam que deve haver um balanço entre tais fatores e que é possível minimizar a latência e ainda garantir uma forte consistência de dados tanto para operações de escrita como leitura.

Em [Kozma and Morschheuser 2019], os autores fazem uma comparação extensa entre os bancos PostgreSQL e Cassandra com foco em latência e número de operações por segundo. Variando o número de nós em *cluster* e as operações realizadas com o YCSB, eles concluem que o Cassandra provê uma maior saída de dados na maioria dos casos, sendo menor a diferença entre os dois bancos na operação de escrita em alguns casos.

O artigo [Barata and Bernardino 2016] faz uma análise minuciosa do banco Cassandra utilizando o YCSB. Para avaliação, são utilizadas diversas configurações do número de nós, número de *threads*, características de *workload* para verificar propriedades de velocidade e escalabilidade. Em [Abramova et al. 2014] é feita uma avaliação semelhante, variando diversos *workloads* do YCSB e verificando o impacto na velocidade, latência e escalabilidade.

A referência [Abramova and Bernardino 2013] faz a comparação de dois bancos de dados *NoSQL* muito utilizados: MongoDB e Cassandra. No artigo, são explicitadas as principais diferenças entre os dois, as *features* de cada e também testes dos *worloads* do YCSB para fins de comparação de resultados. Para a maioria dos cenários, o Cassandra apresenta melhor performance que o MongoDB.

No trabalho presente, a maior diferença está na forma que os testes foram realizados. Para grande parte das referências citadas, os testes foram feitos utilizando a ferramenta YCSB para diversos *workloads*. Neste artigo, executamos os experimentos utilizando dados públicos abertos sobre o Exame Nacional do Ensino Médio (Enem) para *workloads* específicos de leitura utilizando serviços de *clustering*, monitoramento e *deployment* de *Virtual Machines* do Google Cloud Platform.

4. Experimento

O experimento realizado nesse trabalho foi baseado em [Abramova et al. 2014]. Como na maioria das outras referências, o trabalho citado usa o YCSB para benchmarking do banco de dados em questão. De acordo com explicação prévia, no trabalho presente foram utilizados dados do Enem com o objetivo de avaliar critérios como quantidade de nós no *cluster*, quantidade de dados no banco, latência de rede e sua influência nos resultados de consulta no Cassandra.

4.1. Infraestrutura

A infraestrutura utilizada para os testes foi disponibilizada pelo Google Cloud Platform. Em um de seus serviços, Marketplace, é possível selecionar soluções prontas para tecnologias ou *frameworks* mais comuns no mercado. Para o teste em questão, foi selecionado o *One-click-deploy Cassandra*, em que um *cluster* é criado e disponibilizado para o cliente de acordo com os requisitos e recursos desejados. As *Virtual Machines* selecionadas, denominadas *e2-highmem-4*, tinham como configuração 4 vCPUs, 32 GB de memória

Ano	Linhas	GB	Arquivo
2016	8.627.367	5,31	microdados_enem_2016.csv
2017	6.731.341	3,78	MICRODADOS_ENEM_2017.csv
2018	5.513.747	3,21	MICRODADOS_ENEM_2018.csv
2019	5.095.270	2,99	MICRODADOS_ENEM_2019.csv
Total	**25.967.725**	**15,29**	

Tabela 1. Quantitativo dos Arquivos de Carga

RAM, com uma unidade de 10 GB destinada para sistema operacional e outra unidade reservada para dados (1 TB). O sistema operacional é o Debian 9.13, com pacote Java JMX-Exporter 0.11.0 e OpenJDK 1.8.0. A versão do Cassandra disponibilizada é a 3.11 (última estável).

Com a finalidade de não onerar um dos nós e também para que não houvesse influências de rede (latência entre zonas), foi adicionada uma VM (dentro do mesmo datacenter, zona e região) para ser *client*, que ficou dedicada à execução das consultas e armazenamento dos resultados. Essa VM (modelo *e2-standard-2*) tinha 2 vCPUs e 8 GB de memória RAM, no mesmo sistema operacional, Debian 9.13.

4.2. Carga dos dados

Os dados do Enem são disponibilizados e foram obtidos através do portal do INEP, órgão responsável pela organização do evento. Divididos em anos, estes foram dispostos em formato CSV juntamente com seu dicionário de dados. Cada arquivo tem em média 6,5 milhões de registros, distribuídos conforme Figura 1, totalizando aproximadamente 26 milhões de registros, e possui toda a informação do respectivo ano consolidado em uma única tabela.

Neste trabalho, foi adotada como premissa a utilização dos dados *as-is*, ou seja, sem intervenção. Caso alguma mudança fosse feita, deveria ser a menor possível. Contudo, isso não foi necessário.

Com as informações dos dicionários de dados, foi construída uma tabela única, correspondente ao descrito nos dicionários, respeitando o nome dos campos e seus tipos de dados. A estrutura das tabelas dos anos escolhidos tem poucas variações, limitando-se a acrescentar alguns campos, não havendo alteração de tipos de dados entre elas. Na prática, a tabela única tem a estrutura do ano de 2016, por ser a que possui mais campos. Essa possui 30 campos a mais que 2019, e 28 a mais que 2017 e 2018. A tabela final resultante tinha, assim, 167 colunas no total.

Os arquivos foram transferidos para um dos nós do *cluster* (escolhido arbitrariamente) e em seguida procedeu-se a carga nos *databases* através do *Dsbulk*, oferecido pelo Data Stax, disponível em [DataStax 2021b]. Trata-se de um programa que foi instalado no mesmo nó para onde foram transferidos os arquivos e, através de linha de comando, foi feita a carga de cada um deles por vez.

Para os testes, foram criados quatro *keyspaces*. O primeiro continha somente os dados do ano de 2019. No segundo, a junção de 2019 e 2018. No terceiro, 2019, 2018 e 2017 e o último dos quatro anos. A Tabela 2 mostra o quantitativo de registro de cada um dos *keyspaces*. Dessa forma, foi possível variar a quantidade de dados no banco para

Keyspace	Registros
bdm19	5.095.271
bdm1918	10.609.019
bdm191817	17.340.361
bdm19181716	25.967.729

Tabela 2. Quantitativo de registros por Keyspace

realizar os testes. Em cada um dos *keyspaces* foi criada uma única tabela chamada *enem*, todas com a mesma estrutura.

Os *keyspaces* foram criados com suas configurações padrão de particionamento, onde a *Class* foi *SimpleStrategy*, que indica que em todos nós terão o mesmo fator de replicação, cujo o valor *replication_factor* foi configurado para ser igual a 1 (os dados são armazenados em apenas uns dos nós). Esse valor é apresentado na documentação como ”padrão para ambientes de testes e desenvolvimento” [DataStax 2021a].

As configurações de consistência de leitura e gravação também tiveram seu valor como 1 (valor padrão). Isso quer dizer que as operações neste *keyspace* aguardam pela confirmação de apenas um nó para serem efetivadas.

4.3. Elaboração das consultas de teste

Uma vez definida a estrutura de uma tabela no Cassandra, no momento da inclusão do registro é aplicada uma função de *hash* sobre a chave primária. O resultado desse *hash* é um número entre -2^{63} e $2^{63} - 1$, chamado *Token*. A função padrão do Cassandra para calcular esse *hash*, também utilizada por nós, é a *Murmur3Partitioner*. Para nossa chave primária adotamos a composição dos campos NU_ANO e NU_INSCRICAO, pois garantem a unicidade.

A configuração *Num_tokens* indica, em cada um dos nós, quantas partes da faixa de *tokens* serão administradas pelo nó. Uma vez configurado o mesmo valor em todos os nós, todos terão o mesmo quantitativo desta faixa de *tokens*. No nosso caso, mantivemos o valor padrão para essa entrada, 256, em todos os nós, o que garante a distribuição equitativa de carga entre eles.

Para geração das consultas de leitura a serem testadas, utilizamos valores obtidos do próprio banco através de amostras. Para isso, foi implementado um *script* Python, que gerava um número aleatório com a função *randint* (biblioteca *random*) dentro da faixa de *tokens* descrita acima. Em seguida fazemos uma consulta ao banco que traz o registro com o número de *token* imediatamente acima desse valor. Desse registro compomos a seleção, utilizando a chave primaria composta NU_ANO e NU_INSCRICAO. Para a projeção tomamos quatro campos (TP_ENSINO, NU_INSCRICAO, CO_MUNICIPIO_PROVA, CO_UF_PROVA) escolhidos arbitrariamente. Por meio dessa abordagem, as amostras de consulta para testes foram selecionadas dentro de uma distribuição normal, garantindo a aleatoriedade das mesmas. Esse processo é comum em testes e é uma das opções presentes na ferramenta YCSB, também se assimilando com o teste modelo utilizado.

4.4. Execução e armazenamento dos testes

Utilizando o processo explicado na sub-seção anterior, foram geradas 10.000 consultas aleatórias e exclusivas para cada *keyspace*. Essas consultas foram executas para cada configuração desejada e são similares ao padrão do *Workload C* do YCSB, que é um teste 100% de leitura.

Como o objetivo era testar a influência no desempenho quando se varia a quantidade de dados e número de nós do *cluster* na execução das *queries*, o processo foi o seguinte: pra cada *keyspace*, foram realizadas 10.000 consultas, variando o número de nós a cada vez. Dessa forma, a quantidade de dados no banco era variada de acordo com o *keyspace* e o número de nós de acordo com a adição de mais instâncias de VMs no *cluster*. Consequentemente, foram realizadas 40.000 consultas nos *keyspaces*, repetidas 4 vezes (de 3 a 6 nós no *cluster*) em cada, totalizando 160.000 consultas em 16 testes diferentes.

Para realização dos testes, foi feito um *script* Python, que estabelecia a conexão com o *cluster*, realizava cada um dos testes, armazenando os resultados em arquivos CSV para posterior análise. Como explicado anteriormente, o código foi executado na VM *client*, implantada na mesma zona e região dos nós do banco para minimizar as latências de rede.

5. Resultados

Nas Figuras 1 e 2 são apresentados os resultados principais do experimento. Nelas são exibidos, respectivamente, os tempos totais de execução das 10.000 consultas e o tempo singular de cada consulta para as configurações testadas.

Para responder a **QP1**, é interessante observar a mudança do tempo ao longo do eixo x (onde ocorre o aumento ou diminuição da quantidade de dados) para barras ou *box-plots* da mesma cor. Para a Figura 1, é possível perceber que para um aumento de aproximadamente 5 vezes o número de dados (5 para 26 milhões), o tempo total de execução aumenta apenas cerca de $3,5\%$ ($17,47s$ para $18,08s$) para execução com 3 nós, ao passo que esse aumento é de cerca de $10,24\%$ ($27,28s$ para $30,07s$) para execução em 6 nós. Na Figura 2, analisando de forma semelhante, verifica-se que a mediana do tempo de execução singular das consultas varia apenas $3,5\%$ ($1,76ms$ para $1,82ms$) para 3 nós e $7,7\%$ ($2,74ms$ para $2,95ms$) para 6 nós, variando a mesma quantidade da dados (5 para 26 milhões). Dessa forma, é possível concluir que a quantidade de dados não influencia fortemente no desempenho de execução das consultas, tornando o Cassandra muito eficiente para aplicações de grande volume.

Para a **QP2**, analisa-se cada agrupamento do eixo x, onde a quantidade da dados é fixa e o que varia é o número de nós no *cluster*. Na Figura 1, para o agrupamento de 5 milhões, há uma grande variação no tempo ($17,47s$ para $26,24s$, $50,28\%$) quando o número de nós aumenta de 3 para 4. Essa variação é menor de 4 para 5 nós ($22,20s$ para $27,12s$, 22,16%). De 5 para 6 nós, a variação é grande novamente ($23,23$ s para $30,07s$, $29,47\%$). Comparando os dois valores extremos, de 3 para 6 nós, verifica-se uma variação muito grande ($17,47s$ para $30,07s$, $72,16\%$). Esse comportamento é parecido quando os outros agrupamentos são analisados. Na Figura 2, os resultados são semelhantes, havendo uma diferença considerável na mediana dos tempos entre os polos. De 3 para 6 nós, por

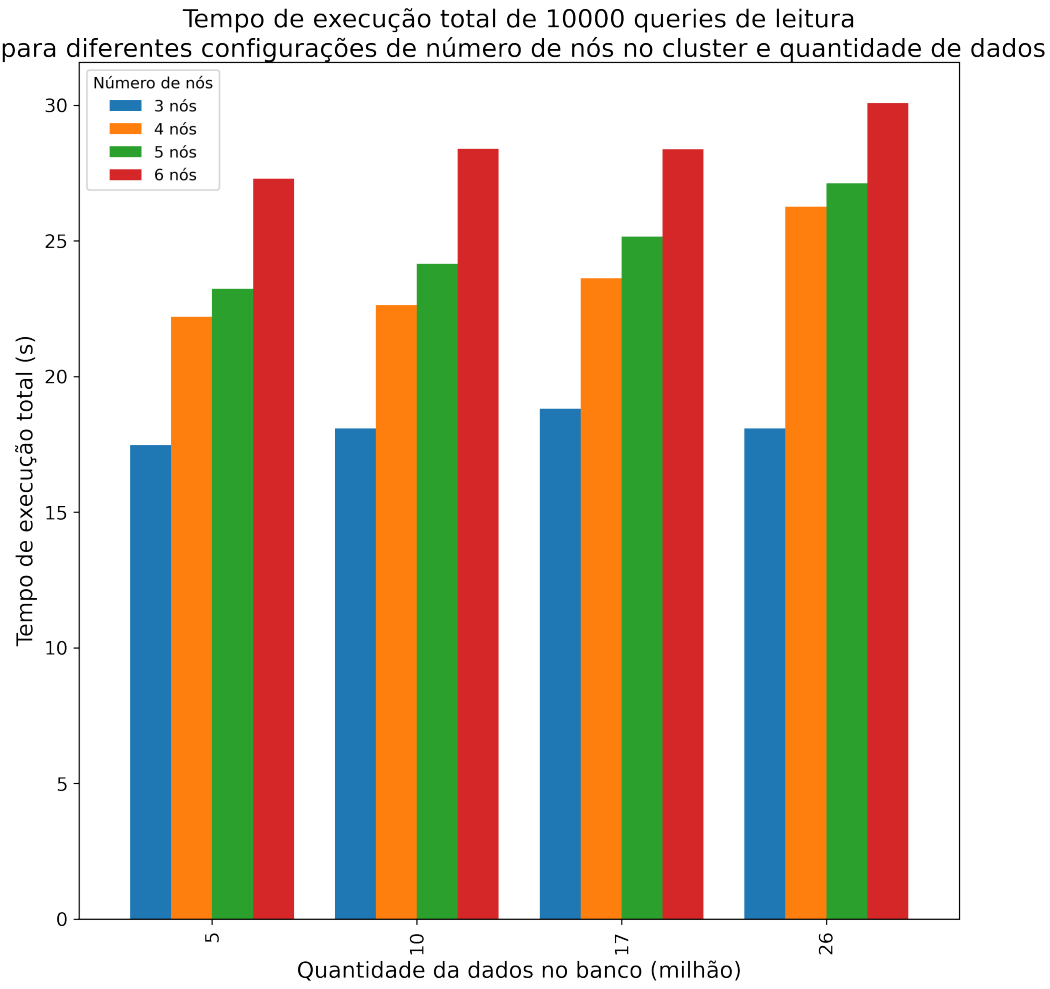

Figura 1. Diagrama de barras para execução dos testes. Os valores no eixo x representam o número de registros em cada *keyspace* onde o teste foi realizado. No eixo y, a soma do tempo de execução para as 10.000 consultas para cada configuração.

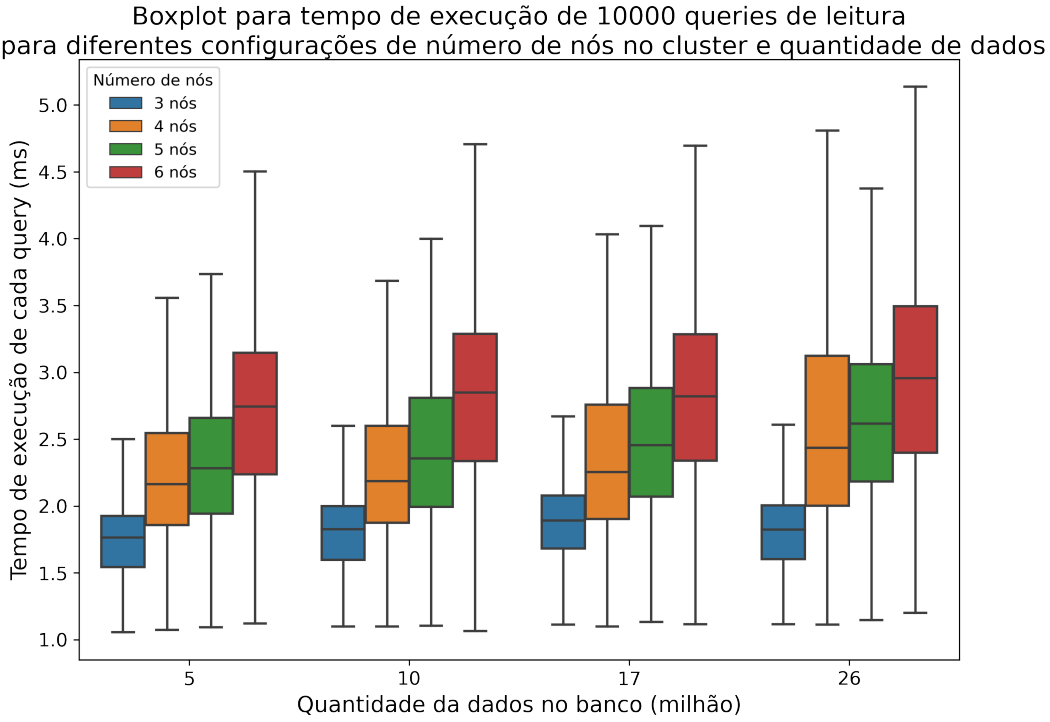

Figura 2. Boxplot para execução dos testes. Os valores no eixo x representam o número de registros em cada *keyspace* onde o teste foi realizado. No eixo y, o tempo de execução para cada uma das consultas para cada configuração.

exemplo, esse valor passa de $1,7ms$ para $2,8ms$, uma variação de $67,61\%$. Nota-se, também, que há uma flutuação maior entre os valores com o aumento de nós. Isso é verificado através do tamanho do *boxplot* e os indicadores de valores mínimo e máximo. Alguns valores muito diferentes, entre $40ms$ e $80ms$, apareceram poucas vezes nos registros. Esses números foram retirados da representação da Figura 2 para não prejudicar a visualização da escala e dos resultados. Tal ocorrido pode ser justificado por picos de rede entre os nós. Respondendo a **QP2**, o aumento do número de nós em um *cluster* não significa, necessariamente, melhoria do tempo de execução das consultas de leitura. Isso é dado pois, apesar da melhoria na distribuição de *index* entre nós, há um aumento considerável nas comunicações de rede entre eles também. Dessa forma, essa pequena latência a cada execução tem impacto nos resultados gerais e deve ser considerada de acordo com a aplicação, fazendo um ajuste fino para balanço nos parâmetros de consistência de dados e latência de rede dependendo do volume e requisitos de projeto.

De forma geral, o Cassandra se provou uma boa opção para banco de dados de grandes volumes. Sua performance não é alterada significativamente com o aumento da quantidade de dados, o que é um ponto importante para essa tarefa. Em relação ao número de nós, tal fator influencia na velocidade das consultas por causa das pequenas comunicações entre eles. Por isso, esse número deve ser considerado mais cautelosamente nos projetos, fazendo um balanço entre consistência, velocidade e latência de rede. Complementando, o Cassandra também dispõe de diversas técnicas e parâmetros que fa-

vorecem na escalabilidade, alta disponibilidade e velocidade de suas tarefas.

6. Conclusão e Trabalhos Futuros

O trabalho presente apresentou um experimento de *benchmarking* do banco Cassandra com o objetivo de verificar a influência da quantidade de dados e número de nós no *cluster* nos resultados de execução de consultas de leitura. Para isso, foram utilizados dados do Enem, exame nacional realizado anualmente para avaliar alunos do Ensino Médio. Os dados foram carregados no banco e testados em diversas configurações para os critérios analisados utilizando *scripts* Python em um *cluster* do Google Cloud Platform.

Após o experimento, foi verificado que o Cassandra escala de forma muito eficiente de acordo com a quantidade de dados, tendo um aumento de apenas $3,5\%$ no tempo total de execução das consultas utilizando 3 nós e $3,5\%$ na mediana do tempo de execução de cada consulta para o mesmo número de nós, quando variada a quantidade de registros. Em relação à influência do número de nós, verificou-se que, para a mesma quantidade de dados no banco, o aumento desse número também acarreta em um maior tempo de execução das consultas devido, principalmente, à latência de rede entre nós. Aumentando de 3 para 6 nós, o tempo de execução total foi acrescido em $72,16\%$ e a mediana de cada execução em $67,61\%$ para um banco com 5 milhões de registros.

Para trabalhos futuros, alguns fatores podem ser explorados. O primeiro deles seria fazer um comparativo dos testes, que utilizaram como consultas apenas chaves primárias, com consultas que utilizassem os demais índices disponibilizados pelo Cassandra (dentre eles a chave clusterizada, índice secundário e secundário composto). As chaves primárias tem relação direta com a distribuição dos dados dentre os nós, ao passo que estas outras tem seus dados de índice armazenados a parte. São formas distintas de tratamento, o que deve interferir nos resultados. Também seria interessante saber como o fator de consistência, aqui definido como *ONE*, pode interferir no desempenho. Este nível é o menos exigente, ao contrário do nível *ALL* que exige que todos os nós respondam a questão. E entre estes dois extremos, como se sairiam seus intermediários (*TWO*, *THREE*, *QUORUM*, *LOCAL_QUORUM* e *EACH_QUORUM*). Outro fator a ser explorado seria o número de *threads* para execução das consultas. Tal parâmetro é explorado em outros trabalhos e também pode influenciar nos resultados.

Referências

Abramova, V. and Bernardino, J. (2013). Nosql databases: Mongodb vs cassandra.

Abramova, V., Bernardino, J., and Furtado, P. (2014). Testing cloud benchmark scalability with cassandra. pages 434–441. Institute of Electrical and Electronics Engineers Inc.

Barata, M. and Bernardino, J. (2016). Cassandra's performance and scalability evaluation.

DataStax (2021a). Create keyspace. `https://docs.datastax.com/en/cql-oss/3.3/cql/cql_reference/cqlCreateKeyspace.html`. Accessed: 2021-11-03.

DataStax (2021b). Datastax bulk loader 1.8.0 for apache cassandra. `https://docs.datastax.com/en/dsbulk/doc/dsbulk/reference/dsbulkCmd.html`. Accessed: 2021-11-03.

Gorbenko, A., Romanovsky, A., and Tarasyuk, O. (2020). Interplaying cassandra nosql consistency and performance: A benchmarking approach. volume 1279 CCIS, pages 168–184. Springer Science and Business Media Deutschland GmbH. estudo dos parametros de consistencia em contraponto com latencia.

INEP (2021). Exame nacional do ensino médio (enem). https://www.gov.br/inep/pt-br/areas-de-atuacao/avaliacao-e-exames-educacionais/enem. Accessed: 2021-11-04.

Kozma, F. and Morschheuser, T. (2019). Cloud service environment postgresql vs. cassandra.

Mansouri, Y. and Babar, M. A. (2020). The impact of distance on performance and scalability of distributed database systems in hybrid clouds. Performance em nuvens hibridas - privada e publica. Foco na influencia da distancia entre clouds.

Silveira, R., Victorino, M., and Holanda, M. (2020). Rolap dw transformation proposal for olap architecture in nosql database. pages 1–7.