Uma Análise de Soluções NewSQL

Ronan R. Knob[1], **Geomar A. Schreiner**[1],
Angelo A. Frozza[1,2] , **Ronaldo dos Santos Mello**[1]

[1]Departamento de Informática e Estatística (INE)
Universidade Federal de Santa Catarina (UFSC)
Florianópolis, SC – Brazil

[2]Instituto Federal Catarinense (IFC) - Campus Camboriu
Rua Joaquim Garcia, S/N – 88.340-055 – Camboriu (SC), Brasil

geomarschreiner@gmail.com, angelo.frozza@ifc.edu.br

ronanknob@grad.ufsc.br, r.mello@ufsc.br

Abstract. *Several applications have as requirements the need to handle large and heterogeneous data volumes as well as the support to handle thousands of OLTP transactions per second. Traditional relational databases (DBRs) are not suitable for these requirements. On the other hand, NoSQL DBs are able to deal with Big Data, but lacks the support to ACID properties. NewSQL is a new class of DBs that combines the support to OLTP transactions of BDRs with the high availability and scalability of NoSQL DBs. However, few works in the literature explore the differences among different NewSQL solutions. In this paper, we execute benchmark software to compare the most prominent NewSQL products analyzing the results. This analysis can be useful as a guide to future use of NewSQL technology.*

Resumo. *Diversas aplicações produzem e manipulam grandes volumes heterogêneos de dados, bem como necessitam lidar com um grande número de transações OLTP. Os tradicionais Bancos de Dados Relacionais (BDRs) não são adequados a este tipo de demanda. Já os BDs NoSQL, apesar do melhor gerenciamento de Big Data, não garantem as propriedades ACID. O movimento NewSQL visa suportar transações OLTP dos BDRs com uma arquitetura distribuída que oferece alta escalabilidade e disponibilidade, típica dos BDs NoSQL. Poucos trabalhos na literatura exploram as diferenças entre soluções NewSQL. Assim, este trabalho visa comparar alguns dos principais produtos NewSQL utilizando benchmarks de domínio. Esta análise contribui como um guia de referência para futuros usos da tecnologia NewSQL.*

1. Introdução

Os avanços em tecnologias Web e a proliferação de dispositivos móveis conectados à Internet gerou uma necessidade de tratamento de grandes quantidades de dados heterogêneos em curto espaço de tempo. Dados com esta natureza de gerenciamento são denominados *Big Data*. Um conjunto de novas aplicações, como sistemas financeiros e jogos *online*, lidam com um grande número de transações *OLTP* (*Online Transaction Processing*) executadas sobre Big Data [Stonebraker 2012]. Transações OLTP são, em

geral, transações de curta duração e que não processam grandes quantidades de dados. BDRs, apesar de empregados na manipulação eficiente de dados durante décadas, não são adequados ao tratamento de Big Data e transações OLTP com alta disponibilidade impostos por estas aplicações [Pavlo and Aslett 2016].

Motivado por esses desafios, surgiram novas soluções de BD denominadas BDs NoSQL (*Not Only SQL*). Estas soluções oferecem recursos como alta disponibilidade e escalabilidade, atrelados a uma arquitetura distribuída e com crescimento horizontal. Apesar de serem capazes de manipular grandes volumes de dados com alta disponibilidade, BDs NoSQL geralmente não suportam as tradicionais propriedades ACID que caracterizam transações OLTP.

Mais recentemente, o paradigma NewSQL surgiu com o propósito de combinar os benefícios do paradigma relacional com o tratamento de Big Data do paradigma NoSQL. Sistemas NewSQL são soluções modernas que buscam prover o mesmo desempenho escalável dos BDs NoSQL para cargas de trabalho OLTP com típico suporte completo a todas as propriedades ACID, como encontrado nos BDRs [Pavlo and Aslett 2016]. Um BD NewSQL deve considerar dois importantes fatores: *(i)* Um controle de concorrência de esquema *lock-free*; e, *(ii)* Uma arquitetura distribuída *shared-nothing* [Stonebraker 2012]. Apesar de compartilharem características gerais, como as citadas anteriormente, cada sistema NewSQL possui sua própria maneira de executar operações de manipulação de dados. Assim sendo, torna-se pertinente uma análise de soluções NewSQL através de *benchmarks* (protocolos de testes padronizados) capazes de mensurar desempenho a fim de verificar a eficácia das mesmas em situações reais.

Este trabalho tem como objetivo contribuir com a literatura acerca do paradigma NewSQL, comparando quatro soluções relacionadas: VoltDB, NuoDB, MemSQL e Cockroach. A comparação é realizada através da execução de dois *benchmarks* (YCSB e Votter), bem como uma análise dos resultados dos testes para estas soluções. A escolha das soluções considerou os seguintes critérios: o ranking das mesmas no site *DB-Engines*[1]; número de menções em Web sites e interesse nas buscas (via Google Trends[2]); a análise da pesquisa anual do Web site *Stack Overflow* no ano de 2017 [3]; e licença de uso gratuito que permitisse os testes. A seleção dos *benchmarks* levou em consideração aqueles que possuíam um foco maior em transações OLTP.

O restante deste artigo está organizado conforme segue. A Seção 2 apresenta os trabalhos relacionados, enquanto a Seção 3 descreve os BDs NewSQL selecionados. O ambiente experimental e os *benchmarks* escolhidos são descritos na Seção 4. Na Seção 5 são discutidos os resultados e, por fim, as considerações finais encontram-se na Seção 6.

2. Trabalhos Relacionados

Poucas iniciativas para comparação exclusiva entre BDs NewSQL são encontradas na literatura. Cabe ressaltar que existem alguns trabalhos que comparam BDs NewSQL e NoSQL [Grolinger et al. 2013, Hajoui et al. 2015, Gurevich 2015]. Porém, eles foram desconsiderados pois realizam uma análise de cunho teórico entre soluções de dois paradigmas diferentes, não comparando unicamente soluções NewSQL.

[1] `https://db-engines.com/en/ranking`
[2] `https://trends.google.com.br/trends/`
[3] `https://insights.stackoverflow.com/survey/2017`

O trabalho de [Pavlo and Aslett 2016] faz uma análise histórica de sistemas de gerência de banco de dados (SGBDs) NewSQL. Porém, o trabalho não possui experimentação. Já os trabalhos de Kaur [Kaur and Sachdeva 2017] e Oliveira [Oliveira and Bernardino 2017] relatam avaliações de desempenho entre produtos NewSQL. O trabalho de Kaur realiza uma comparação entre os SGBDs *NuoDB*, *Cockroach*, *VoltDB* e *MemSQL* utilizando métricas simples para avaliação que podem ajudar no processo de escolha de uma solução. Entretanto, elas não são métricas adequadas, pois não consideram um ambiente distribuído no qual soluções NewSQL são tipicamente utilizadas. Já o trabalho proposto por Oliveira avalia o desempenho dos SGBDs *VoltDB* e *MemSQL* utilizando o software de benchmark TPC-H. O TPC-H, porém, tem como foco transações OLAP, enquanto soluções NewSQL priorizam em seu desenvolvimento transações OLTP. Além disso, o trabalho de Oliveira também não leva em consideração um ambiente distribuído para realizar seus experimentos.

O trabalho proposto neste artigo se difere dos trabalhos de Kaur e Oliveira ao comparar soluções *NewSQL* utilizando dois softwares de *benchmark* focados em transações OLTP em um ambiente distribuído.

3. Bancos de Dados NewSQL

BDs NewSQL são uma nova classe de BD que oferecem o mesmo desempenho escalável dos BDs NoSQL enquanto garantem as tradicionais propriedades ACID dos BDRs [Pavlo and Aslett 2016]. Eles são geralmente BDs em memória principal que maximizam a vazão de dados, prevenindo custosos acessos em disco aos dados. Eles também possuem mecanismos de controle de concorrência que evitam o bloqueio de dados, possibilitando alta disponibilidade de dados. Além disso, são BDs nativamente distribuídos com arquiteturas e algoritmos otimizados para este ambiente. Apesar destas características em comum, cada solução NewSQL possui, evidentemente, uma implementação distinta. As subseções a seguir apresentam brevemente os quatro SGBDs NewSQL selecionados neste trabalho. Uma comparação mais detalhada entre as soluções pode ser encontrada em [Knob 2018].

3.1. VoltDB

O VoltDB[4] é um SGBD desenvolvido desde 2010 por uma empresa que carrega o mesmo nome. Ele é disponibilizado em versões *enterprise* e *community*, sendo esta última sob licença *GNU Affero General Public License* [VoltDB 2015].

O VoltDB possui grande ganho de desempenho por serializar o acesso a todos os dados, prevenindo o consumo de tempo de funções de *latching* e *logs* de transação, dentre outras. Ele possui uma arquitetura de *cluster* com replicação sob múltiplos servidores, que garantem escalabilidade, confiabilidade e alta disponibilidade dos dados. O VoltDB é compatível com a linguagem *SQL ANSI*, o que garante uma rápida curva de aprendizado dos usuários.

Na arquitetura do VoltDB, os *clusters* contêm uma fila de processamento, uma *engine* de execução e as tabelas com os dados indexados. A comunicação entre nodos é realizada quando há necessidade de processar uma consulta que necessita dados de

[4] `https://www.voltdb.com/`

múltiplas partições. Neste cenário, um dos nodos age como coordenador e distribui o trabalho necessário entre os demais nodos, coletando os resultados e completando a tarefa [VoltDB 2013].

3.2. NuoDB

O NuoDB[5] lançou sua primeira versão em 2014. O produto é disponibilizado sob licença proprietária e possui as versões *Community*, que é gratuita, *Professional* e *Enterprise*. A versão *Community*, utilizada no trabalho, inclui restrições de escalabilidade. A arquitetura do NuoDB na versão gratuita permite uma (1) instância de administração, um (1) *SM (Storage Management)* e três *TEs (Transaction Processsing)*. Já a versão paga garante ilimitados SMs e TEs. Os conceitos de SM e TE fazem parte da arquitetura do NuoDB e são explicados a seguir.

TE é a camada que recebe requisições SQL, sendo constituída de nodos em memória chamados *Transaction Engines (TENs)*. Quando uma aplicação faz requisições ao NuoDB, os TENs criam *caches* em memória para a carga de trabalho da aplicação. As requisições para arquivos que não estão em *cache* são alimentadas com *caches* de memória de outros TENs ou pela camada de gerenciamento de armazenamento. Já um SM é um nodo de processamento que possui componentes em memória e em disco rígido. Ele também oferece garantias de durabilidade dos dados. Múltiplos SMs podem ser usados para aumentar a redundância de dados.

O NuoDB torna-se escalável pela simplicidade de suas duas camadas. As capacidades do cluster são dimensionadas através dos TEs e SMs. Com essa modularidade, há também a opção de escolher um modo de replicação para cada novo BD criado.

3.3. CockroachDB

O CockroachDB[6] foi lançado em 2015. O projeto foi criado para ser um BD *open source* e distribuído, de forma que uma instância pode ser levantada em um computador pessoal comum e ajudar no processamento de requisições [Labs 2018]. O produto é distribuído nas versões *Core* e *Enterprise*, sendo a primeira gratuita. Diferente das demais soluções NewSQL, ele não utiliza armazenamento final em memória principal. Ao invés disso, é feito o aproveitamento de uma estrutura de *clocks* atômicos para escrita de blocos, que facilita o suporte às características ACID nas transações.

A arquitetura do CockroachDB converte comandos SQL em estruturas de dados *Key-Value (KV)*, que são extremamente rápidas de manipular. Essa arquitetura é composta por um *SQL Layer* que recebe as consultas via uma interface API. A camada SQL repassa a consulta para a camada *Transaction Layer*. Esta camada gera um plano de execução para as requisições SQL. O plano é passado para a camada *Distribution Layer*, responsável por manter um mapa com os pares de chaves *KV*, um repositório que descreve todos os dados do *cluster* e sua localização. O mapa é dividido e distribuído em intervalos, de modo que a consulta às chaves não fique centralizada em um nodo.

O armazenamento efetivo dos dados é realizado pelo *Storage Layer*. Cada instância do BD deve possuir ao menos um *store*, que é o espaço no qual o processo

[5] `https://www.nuodb.com/`
[6] `https://www.cockroachlabs.com/`

do BD lê e escreve dados no disco. Os dados são guardados e manipulados através da API *RocksDB*[7], que mantém os pares KV no disco.

3.4. MemSQL

O MemSQL[8] teve sua primeira versão lançada em 2013. Ele é distribuído em duas versões. A *Developer*, gratuita, não é recomendada para uso em produção e tem recursos limitados. Já a versão *Enterprise* (paga) possui todas as funcionalidades. O MemSQL se categoriza como um BDR distribuído que suporta transações e análises em tempo real.

A estrutura do MemSQL é composta de duas camadas: os *nodos agregadores* e os *nodos folha* [MemSQL 2018]. Os nodos agregadores funcionam como roteadores de consulta e atuam como um *gateway* no sistema distribuído. Eles armazenam apenas metadados e dados de referência, distribuindo as consultas nos nós folha. Estes nodos também são responsáveis por agregar os resultados a serem enviados de volta ao cliente. Já os nodos folha armazenam e computam as tarefas. Os dados são automaticamente distribuídos através dos nodos folha, em partições sobre as quais as consultas são executadas de forma paralela.

A comunicação entre os nodos é realizada via comandos SQL sob um protocolo MySQL. A proporção de nodos agregadores e nodos folha determina a capacidade e o desempenho do cluster, que pode variar conforme a aplicação.

4. Ambiente Experimental

Os experimentos propostos para as soluções NewSQL escolhidas necessitaram de uma infraestrutura padronizada, bem como uma configuração de *cluster* específica para cada uma delas. Todas elas foram instaladas e configuradas de maneira padrão (sem nenhuma otimização) em um *cluster* com três nodos físicos. Cada um dos nodos possui um processador Intel® Core™ i5-7200 (4 núcleos físicos de $2,50$ GHz), com Memória RAM 8GB (DDR3 1333Mhz), disco rígido de 320GB (5400 RPM) e sistema operacional Xubuntu 16.04 Server LTS 64 bits. Devido a problemas de configuração, o *VoltDB* foi instalado no *cluster* utilizando *Docker*. Já os demais produtos foram instalados diretamente na máquina. Vale ressaltar que, apesar de todos os produtos terem sido instalados nas mesmas três máquinas, apenas um deles estava ativo durante os testes. Os nodos do *cluster* foram conectados via *Ethernet* (100 Mbps) sem acesso à rede externa.

Os experimentos foram realizados com o uso da ferramenta *OLTP-Bench* [Difallah et al. 2013], que foi executada em um nodo externo ao *cluster*. A *OLTP-Bench* é uma suíte de *benchmarking* que traz suporte a vários SGBDs comerciais, assim como uma boa carga de *benchmarks* distintos. Basicamente, a ferramenta gera uma fila de transações para execução de acordo com a especificação do *benchmark* escolhido e um arquivo de configuração fornecido pelo usuário. Esta fila é executada paralelamente por um número de *workers* (que emulam usuários) configurado no arquivo de entrada. Durante a execução dos testes, o *OLTP-Bench* coleta as estatísticas de execução retornando um arquivo com esses valores.

Considerando as características dos BDs NewSQL, foram selecionados *benchmarks* para BDRs com foco em OLTP, ou seja, capazes de simular um cenário com

[7] `https://rocksdb.org/`
[8] `https://www.memsql.com/`

transações simples, porém, em grande quantidade. Os *benchmarks Yahoo! Cloud Serving Benchmark (YCSB)* e *Voter* foram escolhidos com base nestes critérios, sendo apresentados nas próximas subseções.

Dentre as métricas disponibilizadas no arquivo de saída da *OLTP-Bench*, foram selecionadas duas para análise neste trabalho: *(i)* a taxa de transações executadas no tempo (*Throughput*); e, *(ii)* a latência das transações, através da análise da média geral das latências, bem como a análise dos percentis 90 e 99. Para o *benchmark* YCSB também foi realizada a análise de média das latências por tipo de transação. O desvio padrão das amostras observadas também foi calculado para ajudar a explicar as observações.

4.1. YCSB

O YCSB é uma aplicação geradora de carga de trabalho e um pacote com cargas padrão que cobrem interessantes partes da avaliação de desempenho, como cargas de leitura e escrita intensa, varredura de tabelas, dentre outras. Cada carga representa um misto de operações de leitura e escrita com diferentes volumes de dados e número de requisições.

A estrutura de teste consiste em uma única tabela, denominada *usertable*, com N campos. Cada registro é identificado por uma chave primária (algo como "user234123") e cada campo é nomeado como *field0*, *field1*, ..., *fieldN*. Os valores dos campos são strings randômicas de caracteres ASCII de tamanho aleatório. Os parâmetros *número de campos* (N) e *fator de escala* (F) são informados *a priori*.

A execução de um teste realiza muitas escolhas aleatórias, como as operações que serão feitas (*Insert, Update, Read* ou *Scan*), qual registro ler ou escrever, e quantos registros examinar. Essas decisões são governadas por distribuições randômicas. Nos parâmetros definidos para a execução são configurados fatores relacionados à escala de volume da base de teste e os dados da carga de trabalho. Neste trabalho foi utilizado o fator de escala 1000, 64 usuários virtuais emulados para manipulação (64 conexões simultâneas) e um limite do teste de 300 segundos. O volume total de dados gerado foi de $18,2$ GB.

4.2. Voter

O *Voter* é um *benchmark* baseado em um software utilizado em um programa de talentos exibido em televisão no Japão e no Canadá. Os usuários ligam para votar no seu candidato favorito. Ao receber uma ligação, a aplicação invoca a transação que atualiza o número total de votos de cada participante. Os votos feitos por cada usuário são armazenados em um BD e possuem um limite máximo configurável. Uma transação em separado é periodicamente invocada para computar os votos totais durante o programa.

Este *benchmark* foi desenvolvido com o intuito de saturar o BD com pequenas transações, todas atualizando um pequeno número de registros. A arquitetura do *Voter* possui três tabelas que guardam dados sobre os candidatos e o usuário que está ligando. Além disso, existem duas *views* que são consultadas para atualizar o status no programa de televisão. Neste trabalho foram utilizados como parâmetros o fator de escala (1000) e o número de usuários virtuais emulados (64). O volume de dados gerado foi de $2,6$ GB.

5. Análise dos Resultados

Esta seção apresenta os resultados obtidos com os *benchmarks* definidos na seção anterior.

5.1. *Benchmark* YCSB

Os resultados obtidos com o *benchmark YCSB* foram coletados a partir de 5 execuções de teste para cada solução NewSQL. O gráfico da Figura 1 mostra os resultados para a primeira métrica: número de transações executadas por segundo.

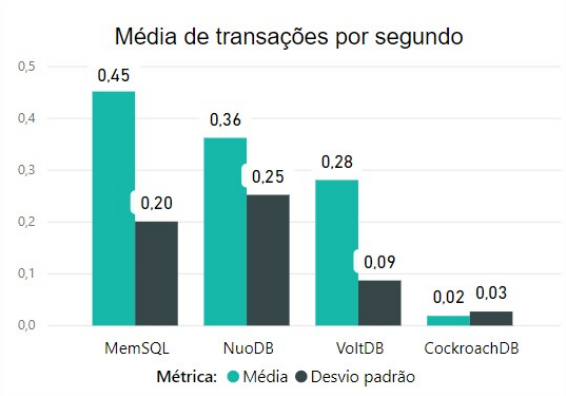

Figura 1. Médias de transações por segundo obtidas no *benchmark YCSB*.

Uma análise do gráfico demonstra que o *MemSQL* apresentou melhores resultados, executando em média $0,45$ transações a cada segundo do teste, à frente do NuoDB, que executou $0,36$. O *VoltDB*, por sua vez, executou $0,28$ transações por segundo em média e, por último, tem-se o *Cockroach*, executando $0,02$ transações por segundo em média. O desvio padrão mostra que o grau de dispersão entre os resultados foi baixo, ou seja, houve uma boa uniformidade na execução. As maiores discrepâncias de execução ficaram por conta do *MemSQL* e do *NuoDB*. A diferença considerável visível no Figura 1 do produto *CockroachDB* para os concorrentes pode ser explicada comparando as demais métricas, apresentadas a seguir.

Conforme descrito anteriormente, outra métrica considerada é a média de latência das transações. Os resultados para esta métrica são mostrados no gráfico da Figura 2.

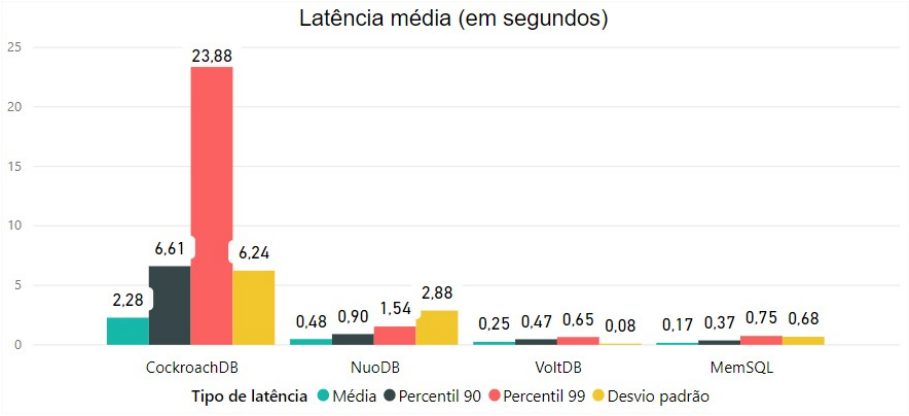

Figura 2. Médias de latência por segundo obtidas no *benchmark YCSB*.

Como se pode observar, o *CockroachDB* também apresentou maior latência média nas transações, quando não se considera o tipo de transação. O desvio padrão mostra que o grau de dispersão entre os resultados foi alto para *CockroachDB* e *NuoDB*, indicando que

estes produtos tiveram uma grande discrepância de valores, diferente dos outros produtos que mostraram uma execução mais uniforme. Em relação à média, *NuoDB* e *VoltDB* sucederam o *CockroachDB* no *ranking*, com resultados parecidos entre si. A melhor latência média ficou por conta do *MemSQL* ($0,17$ por segundo).

Uma análise dos percentis 90 e 99 mostra que uma quantidade expressiva das transações com maior latência se encontra em um pequeno número de transações situadas nesses pontos. O caso mais evidente é o *CockroachDB* (Figura 2), para o qual há uma grande diferença entre a média de $2,28$ segundos na média de latências e $23,88$ segundos na média de latências no percentil 99.

Os resultados apresentados pelo *CockroachDB* para latência média e média de transações por segundo são discrepantes com relação aos demais produtos. Isto se deve ao fato que o armazenamento principal do CockroachDB não é em memória principal, como os demais produtos, e sim em estruturas *KV* em disco. Este BD utiliza um percentual da memória principal para *cache*, mas utiliza grande parte de suas operações baseadas em disco, o que gerou as latências e a taxa de vazão observadas.

Discrepâncias também são notadas para o *NuoDB*. Tanto na média de transações por segundo, quanto na análise de latência média, o produto mostra um grau de dispersão um pouco elevado. Tais resultados tendem a ocorrer tendo em vista as limitações da versão gratuita, na qual é possível utilizar o armazenamento em apenas um dos nodos do *cluster*, mesmo que seja possível utilizar o componente de execução de transações em três nodos. O armazenamento em um *storage* único pode aumentar a latência da execução, visto que os nodos executam parte do teste em um nodo, transferindo todo o resultado para o nodo que mantém os dados.

5.2. *Benchmark* Voter

Da mesma forma que o *YCSB*, os resultados no *benchmark Voter* foram coletados com dados de 5 execuções do teste para cada solução. O gráfico apresentado na Figura 3 mostra o resultado da primeira métrica: número de transações executadas por segundo.

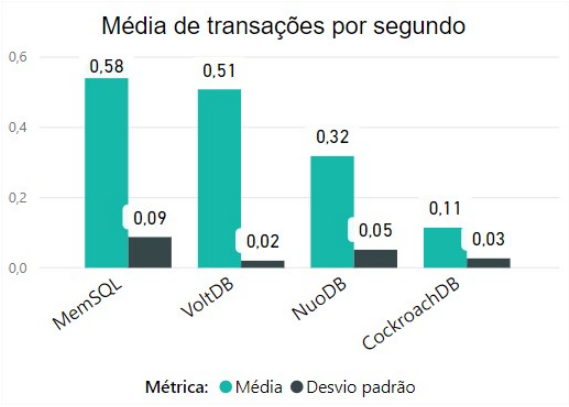

Figura 3. Médias de transações por segundo obtidas no *benchmark Voter*.

Uma análise do gráfico mostra novamente que o *MemSQL* apresenta os melhores resultados, executando em média $0,58$ transações a cada segundo do teste, seguido do *VoltDB*, que executou $0,51$ transações por segundo. O *NuoDB* apresentou um tempo

intermediário, restando novamente ao *CockroachDB* a última colocação. O desvio padrão mostra que o grau de dispersão entre os resultados foi muito baixo, evidenciando uma boa uniformidade na execução.

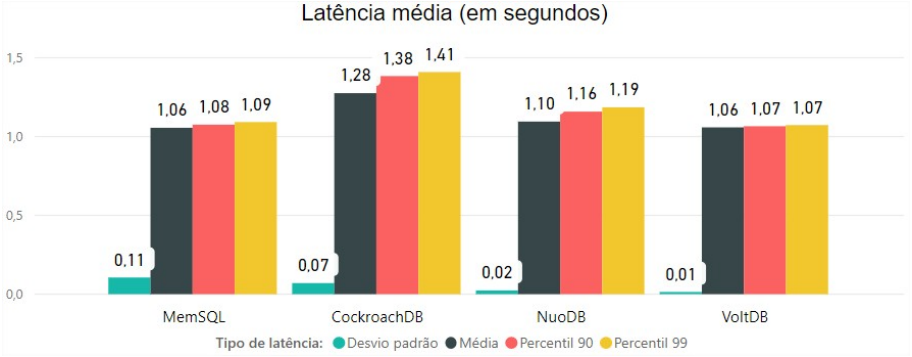

Figura 4. Médias de latência por segundo obtidas no *benchmark Voter*.

O gráfico da Figura 4 apresenta a média das latências das transações obtidas para o *Voter*. Observa-se novamente uma maior latência obtida pelo *CockroachDB*. O desvio padrão mostra que o grau de dispersão entre os resultados foi muito baixo, mostrando uma boa uniformidade na execução. O teste do *Voter* mostra uma situação muito diferente da observada com o mesmo teste realizado com o *YCSB* (Figura 2). No *Voter*, a variação obtida entre um produto e outro é pequena, tanto nas média quanto nos percentis 90 e 99. Isso demonstra que, em transações menores, como é o caso deste *benchmark*, os produtos operam de forma semelhante.

O *Voter* visa uma exploração mais acentuada das características que se esperam de um BD *NewSQL*. Como comentado anteriormente, suas transações consistem em uma votação por número de telefone em um participante do programa *American Idol*. Assim, este *benchmark* acarreta um estresse maior sobre os produtos testados, simulando diversos usuários votando em seu participante favorito. Os BDs *MemSQL* e *VoltDB* apresentaram resultados muito semelhantes, demostrando que são capazes de lidar com várias requisições simultâneas com baixa latência. O *NuoDB* apresenta uma arquitetura com níveis mais complexos de armazenamento, o que prejudica um pouco seu desempenho com um grande volume de transações mais simples. Já o *Cockroach* apresentou o pior resultado. Sua arquitetura, apesar de trazer novos algoritmos, ainda fica limitada a algumas operações que utilizam o disco, depreciando seu desempenho em relação aos demais.

6. Conclusão

Este trabalho analisa quatro produtos que se baseiam no paradigma NewSQL. A técnica de *benchmark* foi empregada, sendo os *benchmarks* escolhidos gerenciados através de um *framework* chamado *OLTP-Bench*. Os benchmarks usados apresentam características específicas para a avaliação de diferentes cenários de ambientes transacionais OLTP. O benchmark *YCSB* é amplamente utilizado para avaliação de BDs distribuídos e possui uma estrutura mais complexa (maior número de tabelas) que envolve uma combinação de transações que vão desde escritas e leituras pesadas à varreduras em tabelas. Já o *Voter*, apresenta uma estrutura mais simples e possui um foco na saturação do BD com diversas requisições rápidas (inserções e atualizações) em um pequeno conjunto de tabelas (3).

Os resultados obtidos revelaram que a solução *MemSQL* se manteve à frente nas características observadas, obtendo alta taxa de *throughput* e baixa latência. Os produtos *VoltDB* e *NuoDB* se comportaram de maneira semelhante na maioria dos resultados analisados, mesmo com as considerações sobre as restrições da versão gratuita do *NuoDB*. O SGBD *CockroachDB* apresentou os piores resultados, com discrepâncias consideráveis nas métricas observadas, principalmente nas taxas médias de transações por segundo.

Como trabalho futuro, pretende-se realizar a mesma análise utilizando particionamento geográfico dos nodos para uma verificação mais aprofundada das características analisadas. Além disso, incluir a avaliação de BDs tradicionais para as mesmas métricas, para comprovar as vantagens que o paradigma *NewSQL* evidencia.

Referências

Difallah, D. E., Pavlo, A., Curino, C., and Cudre-Mauroux, P. (2013). Oltp-bench: An extensible testbed for benchmarking relational databases. *Proc. VLDB Endow.*, 7(4).

Grolinger, K., Higashino, W. A., Tiwari, A., and Capretz, M. A. (2013). Data management in cloud environments: Nosql and newsql data stores. *JoCCASA*.

Gurevich, Y. (2015). *Comparative Survey of NoSQL/NewSQL DB Systems*. PhD thesis, The Open University.

Hajoui, O., Dehbi, R., Talea, M., and Batouta, Z. I. (2015). An advanced comparative study of the most promising nosql and newsql databases with a multi-criteria analysis method. *Journal of Theoretical & Applied Information Technology*, 81(3).

Kaur, K. and Sachdeva, M. (2017). Performance evaluation of newsql databases. In *2017 International Conference on Inventive Systems and Control (ICISC)*, pages 1–5.

Knob, R. R. (2018). Análise e benchmarking das soluções newsql cockroachdb, memsql, nuodb e voltdb. TCC, Universidade Federal de Santa Catarina.

Labs, C. (2018). Architecture overview. `https://www.cockroachlabs.com/docs/stable/architecture/overview.html#goals-of-cockroachdb` Último acesso em: 21/06/2018.

MemSQL (2018). Memsql architecture: Technology innovations power convergence of transactions and analytics. `https://www.memsql.com/content/architecture/` Último acesso em: 06/10/2018.

Oliveira, J. and Bernardino, J. (2017). Newsql databases-memsql and voltdb experimental evaluation. In *KEOD*, pages 276–281.

Pavlo, A. and Aslett, M. (2016). What's really new with newsql? *SIGMOD Rec.*, 45(2).

Stonebraker, M. (2012). Newsql: An alternative to nosql and old sql for new oltp apps. *Communications of the ACM. Retrieved*, pages 07–06.

VoltDB (2013). Using voltdb. `http://downloads.voltdb.com/documentation/UsingVoltDB.pdf` Último acesso em: 05/06/2018.

VoltDB (2015). Voltdb technical overview. `http://www.odbms.org/wp-content/uploads/2013/11/VoltDBTechnicalOverview.pdf` Último acesso em: 05/06/2018.