ANÁLISE COMPARATIVA DE BANCO DE DADOS RELACIONAL E NÃO RELACIONAL: AVALIAÇÃO DE DESEMPENHO POR STRESS E LOAD TESTING

Leonardo dos Santos Maier[1], Luciano Antunes[2]

Resumo: O crescimento das aplicações distribuídas e do volume de dados torna desafiadora a escolha entre bancos relacionais e não relacionais, especialmente sob diferentes cargas e níveis de concorrência. Embora haja ampla literatura sobre desempenho, poucos estudos comparam diretamente PostgreSQL e MongoDB em cenários realistas de estresse. Este trabalho preenche essa lacuna por meio de uma análise comparativa com uso do Apache JMeter, simulando cargas de inserção, leitura, atualização e mista, com 100, 200 e 400 threads. As métricas avaliadas foram tempo de resposta, transações por segundo e uso de CPU e memória. Os resultados mostram que o PostgreSQL se destaca em leitura e cargas mistas moderadas, enquanto o MongoDB é mais eficiente em inserções sob alta concorrência. Conclui-se que a escolha da tecnologia deve considerar o perfil de carga e as demandas de escalabilidade do sistema.

Palavras-chave: banco de dados relacional; banco de dados não relacional; PostgreSQL; MongoDB; testes de desempenho; carga concorrente.

[1]Curso de Ciência da Computação, Universidade do Extremo Sul Catarinense (Unesc), leonarmaier@gmail.com

[2]Curso de Ciência da Computação, Universidade do Extremo Sul Catarinense (Unesc), luc@unesc.net

ABSTRACT: The growth of distributed applications and data volume poses challenges in choosing between relational and non-relational databases, especially under varying workloads and concurrency levels. Although the literature on database performance is extensive, few studies offer direct comparisons between PostgreSQL and MongoDB under realistic stress scenarios. This work addresses this gap through a comparative analysis using Apache JMeter to simulate insert, read, update, and mixed workloads with 100, 200, and 400 threads. Metrics such as response time, transactions per second, and CPU and memory usage were evaluated. Results show that PostgreSQL performs better in read operations and moderate mixed loads, while MongoDB is more efficient in insert-heavy scenarios under high concurrency. The findings suggest that the choice of database technology should consider workload profile and scalability requirements.

Keywords: relational database; non-relational database; PostgreSQL; MongoDB; performance testing; concurrency

1 INTRODUÇÃO

O crescimento de aplicações web modernas, como redes sociais e plataformas de e-commerce, exige o gerenciamento eficiente de grandes volumes de dados e de acessos simultâneos. Essa demanda crescente compromete a escalabilidade e o desempenho das soluções tradicionais de armazenamento (Yedilkhan et al., 2023, tradução nossa).

Para enfrentar esses desafios, o modelo de banco de dados desempenha um papel fundamental, pois define a estrutura lógica para organização e manipulação dos dados. Diversos modelos foram propostos ao longo do tempo, como o modelo em rede, hierárquico, relacional e orientado a objetos. Dentre esses, o banco de dados relacional (RDBMS) se consolidou como a solução predominante para gerenciamento de dados, sendo amplamente adotado para armazenar, manipular e recuperar informações. Durante décadas, esse modelo tem sido a principal alternativa na indústria de Tecnologia da Informação (Hassan, 2021, tradução nossa).

Segundo a Oracle (2025), o modelo relacional padronizou a representação e consulta de dados por meio de tabelas, oferecendo uma estrutura intuitiva e eficiente. Outro marco importante foi a adoção da linguagem SQL, baseada na álgebra relacional, que proporciona consultas formais e precisas, contribuindo para o alto desempenho das operações em banco de dados.

Apesar dessas vantagens, o crescimento exponencial dos dados trouxe novos desafios. Os bancos relacionais passaram a enfrentar limitações de desempenho e escalabilidade. Inicialmente, ampliou-se o hardware, mas essa abordagem tem limites físicos, exigindo a distribuição dos dados entre servidores. Como foram projetados para operar em um único nó, esse processo se torna complexo e ineficiente (Corbellini et al., 2017, tradução nossa). Makris et al. (2021) compararam PostgreSQL e MongoDB com dados temporais e observaram que, apesar do bom desempenho do PostgreSQL em consultas gerais, o MongoDB se destacou em operações como interseções espaciais com índices. Isso evidencia que modelos relacionais não atendem igualmente bem a todos os cenários, especialmente os que exigem maior flexibilidade e escalabilidade.

Essas limitações impulsionaram o uso de soluções mais adaptáveis, como os bancos NoSQL, que se destacam por sua escalabilidade horizontal, baixa latência e suporte a modelos de dados variados. Além disso, certos mecanismos do modelo relacional, como o uso de *locks*, podem comprometer o desempenho de sistemas altamente concorrentes (Florencio et al., 2017, tradução nossa).

Com o aumento do volume de dados, a demanda por ferramentas e tecnologias mais poderosas e adaptáveis a esse novo paradigma também cresce. Novas tecnologias de armazenamento, como os bancos de dados NoSQL, surgem como uma opção mais adaptável ao cenário do Big Data (Silva; Lima, 2023, tradução nossa).

De acordo com a Microsoft (2025), os bancos de dados NoSQL, também chamados de "não relacionais", diferem dos bancos de dados tradicionais ao processar grandes volumes de dados não estruturados e em constante mudança. Esses bancos de dados existem desde os anos 60 sob diferentes denominações, mas ganharam popularidade recentemente devido à crescente necessidade de armazenar e processar dados gerados na nuvem, em dispositivos móveis, redes sociais e aplicações de Big Data.

Nesse contexto, Khan et al. (2023) realizaram uma revisão abrangente sobre bancos de dados SQL e NoSQL no ambiente de nuvens computacionais. Os autores discutem aspectos como portabilidade e interoperabilidade entre sistemas e apontam que bancos de dados NoSQL — como o MongoDB — são mais vantajosos em cenários que exigem escalabilidade horizontal, alta disponibilidade e processamento de grandes volumes de dados, enquanto bancos relacionais, como o Oracle, ainda se destacam na consistência transacional e estruturação formal dos dados. Tais obser-

vações reforçam a importância de se avaliar cada modelo de acordo com o tipo de aplicação e carga de trabalho envolvida.

Os bancos NoSQL podem ser classificados em quatro categorias principais: chave-valor, documento, coluna e grafo. Cada tipo possui particularidades e aplicações distintas. Por exemplo, bancos orientados a grafos representam dados em nós e suas relações, otimizando o desempenho em contextos com forte interconectividade (Khan et al., 2023, tradução nossa).

Dentre os bancos de dados NoSQL, o MongoDB se destaca como um sistema de gerenciamento orientado a documentos, oferecendo uma abordagem distinta para o armazenamento de dados. Diferentemente dos bancos de dados relacionais tradicionais, que organizam os dados em tabelas, o MongoDB armazena as informações no formato de documentos, tornando-o especialmente adequado para esquemas de dados complexos ou não padronizados. Esses documentos são representados no formato BSON (Binary JSON), permitindo a acomodação de uma ampla variedade de tipos de dados, incluindo strings, números e arrays (Nuriev et al., 2024, tradução nossa).

Estudos como os de Costa et al. (2022) e Figueiredo et al. (2022) também evidenciam como as características intrínsecas de cada modelo impactam no desempenho. Costa et al., ao analisarem o uso de criptografia no PostgreSQL e no MongoDB, mostraram que a configuração do ambiente e o tipo de operação (como inserções) podem favorecer um ou outro sistema. Já Figueiredo et al. reforçam a relevância do cenário de uso ao compararem diferentes soluções — PostgreSQL, MongoDB e HarperDB — em tarefas de inserção de dados. Mesmo sem o uso de otimizações como índices, os autores destacam como a escolha do banco pode variar conforme as necessidades específicas da aplicação.

O *load testing* é uma técnica essencial no campo de testes de desempenho, voltada para avaliar como um sistema responde quando submetido a diferentes níveis de carga simultânea. Essa abordagem simula acessos concorrentes de múltiplos usuários e permite medir métricas como tempo de resposta, throughput e uso de recursos. Os testes de carga são utilizados para identificar gargalos críticos em aplicações sob uso intensivo, revelando, por exemplo, lentidão em consultas SQL e saturação de conexões, problemas que muitas vezes não são perceptíveis em ambientes com pouca demanda (Pargaonkar, 2023, tradução nossa).

O *stress testing* avalia a robustez do sistema sob condições ex-

tremas, expondo falhas de desempenho, indisponibilidade e problemas de recuperação após sobrecarga. Em bancos de dados, é crucial para analisar o comportamento sob alta concorrência e volumes extremos, permitindo verificar integridade e consistência transacional (Pradeep; Sharma, 2019, tradução nossa). Yenugula, Kodam e He (2019) demonstram que diferentes cargas afetam diretamente o tempo de resposta e a eficiência de sistemas distribuídos. Enquanto o teste de carga identifica gargalos em operações críticas, o de estresse evidencia limitações nos mecanismos de conexão e memória, variando conforme a configuração e uso de índices.

O objetivo deste estudo é comparar o desempenho dos bancos de dados PostgreSQL (relacional) e MongoDB (não relacional) sob diferentes perfis de carga — inserção, leitura, atualização e mista — utilizando as técnicas de load testing e stress testing para simular cenários realistas. A proposta é avaliar a escalabilidade, robustez e eficiência de cada modelo em contextos de alta concorrência. Os resultados obtidos oferecem subsídios técnicos concretos para orientar a escolha da tecnologia mais adequada conforme o perfil de uso e os requisitos da aplicação.

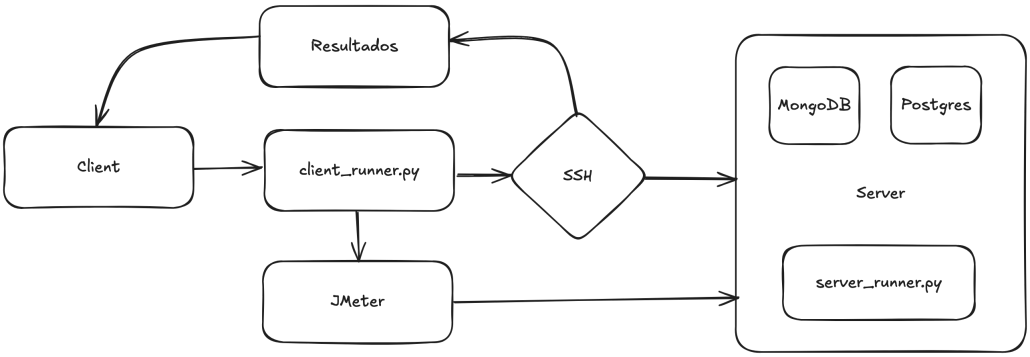
2 MATERIAIS E MÉTODOS

Para avaliar comparativamente o desempenho entre PostgreSQL e MongoDB, foram desenvolvidos planos de teste específicos utilizando a ferramenta Apache JMeter, organizados em quatro perfis de carga: inserção, leitura, atualização e mista. Cada plano foi criado individualmente para cada banco, com equivalência entre as operações e arquivos .jmx organizados em pastas separadas (test-plans/postgres e test-plans/mongo). As operações simuladas refletem ações comuns em aplicações reais, incluindo inserção e atualização de logs, remoção de registros antigos, filtros por nível de log (INFO, ERROR, WARN), contagens agrupadas, recuperação de logs recentes e identificação da fonte mais frequente. Essas queries foram alocadas nos cenários conforme sua natureza, permitindo avaliar o comportamento dos bancos sob diferentes perfis de uso.

Conforme ilustrado na Figura 1, o processo de execução dos testes foi estruturado em uma arquitetura cliente-servidor com automação de ponta a ponta. No lado do cliente, localizado na máquina local, o script `client_runner.py` é responsável por acionar o JMeter, que executa os testes de carga conforme os cenários definidos. Além disso, esse mesmo script estabelece uma conexão remota via SSH com o servidor, onde aciona o script `server_runner.py`. Esse segundo script é responsável por pre-

parar o ambiente de testes no servidor, incluindo a inicialização dos bancos de dados PostgreSQL e MongoDB, bem como a coleta contínua de métricas como uso de CPU e memória durante a execução. Todos os resultados gerados no servidor são enviados de volta ao cliente para posterior análise. Esses dados foram organizados e visualizados por meio de notebooks Jupyter, como `metrics.ipynb` e `resources-charts.ipynb`, os quais possibilitaram a criação dos gráficos apresentados neste trabalho.

Figura 1 - Arquitetura da aplicação



Fonte: Elaborado pelo autor.

O ambiente de execução foi composto por um servidor dedicado com processador AMD Ryzen 5 3600 (6 núcleos e 12 threads), 64 GB de memória RAM DDR4, dois SSDs NVMe de 512 GB e sistema operacional Ubuntu. As ferramentas utilizadas incluíram o Apache JMeter para simulação de carga, Python 3 para automação e coleta de métricas, JupyterLab para análise gráfica, além do pgAdmin e MongoDB Compass para gerenciamento dos bancos.

Os testes foram executados com ramp-up controlado e duração de 1800 segundos por cenário, com 100, 200 e 400 threads concorrentes. Otimizações como índices e ajustes de configuração foram aplicadas a ambos os bancos para simular um ambiente produtivo realista. A análise final considerou métricas como tempo médio de resposta, throughput, distribuição por faixas de latência e uso de recursos computacionais.

Tanto no PostgreSQL quanto no MongoDB, os dados foram modelados com base em registros de log compostos pelos campos: `timestamp` (TIMESTAMP / Date), `log_level` (VARCHAR / String), `message` (TEXT / String), `source` (VARCHAR / String), `test_id` (BIGINT / Int64) e um identificador único (`id` como BIGSERIAL no PostgreSQL e `_id` como ObjectId no MongoDB). Essa equivalência garantiu paridade lógica entre os testes, permitindo comparações justas de desempenho em diferentes operações.

3 DISCUSSÃO E RESULTADOS

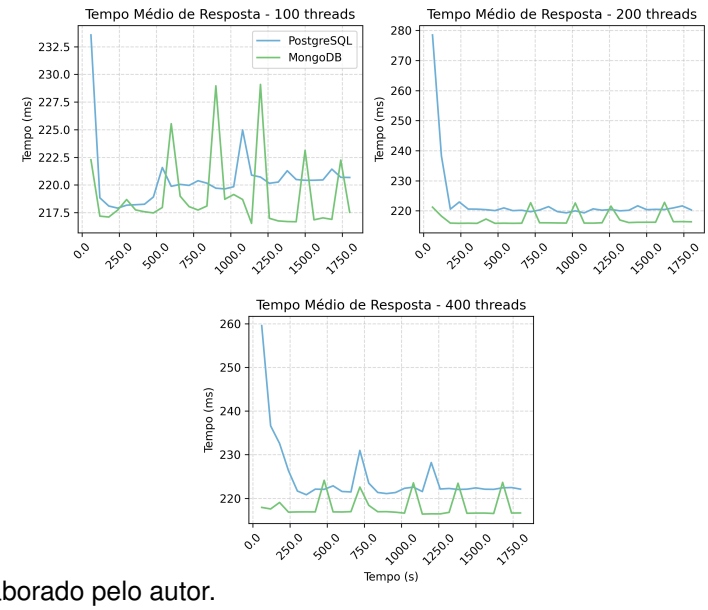
3.1 CARGA DE INSERÇÃO (*INSERT-HEAVY WORKLOAD*)

A Figura 2 apresenta a evolução do tempo médio de resposta, em milissegundos, durante a execução da carga de inserção sob três níveis de concorrência: 100, 200 e 400 threads. Em todos os cenários, MongoDB e PostgreSQL mantiveram desempenhos próximos, com tempos oscilando entre 217 ms e 233 ms. No cenário com 100 threads, o PostgreSQL demonstrou menor variação e comportamento mais linear, enquanto o MongoDB apresentou oscilações pontuais mais acentuadas.

A partir de 200 threads, as diferenças tornam-se menos expressivas. O PostgreSQL tem um pico inicial e em seguida estabiliza com pequenas flutuações, enquanto o MongoDB mantém-se levemente abaixo na média. Em 400 threads, ambos os bancos apresentam tempos similares, mas o PostgreSQL exibe uma maior dispersão em alguns pontos, o que sugere sensibilidade à concorrência elevada.

De forma geral, ambos os bancos mostraram capacidade de manter o tempo médio de resposta dentro de uma faixa estável e adequada, mesmo com o aumento da concorrência. Essa estabilidade reforça os achados de Figueiredo et al. (2022), que destacam a eficiência do MongoDB para operações de escrita distribuída, embora neste caso o PostgreSQL tenha demonstrado competitividade mesmo em altos volumes de inserções simultâneas.

Figura 2 - Tempo médio de resposta.



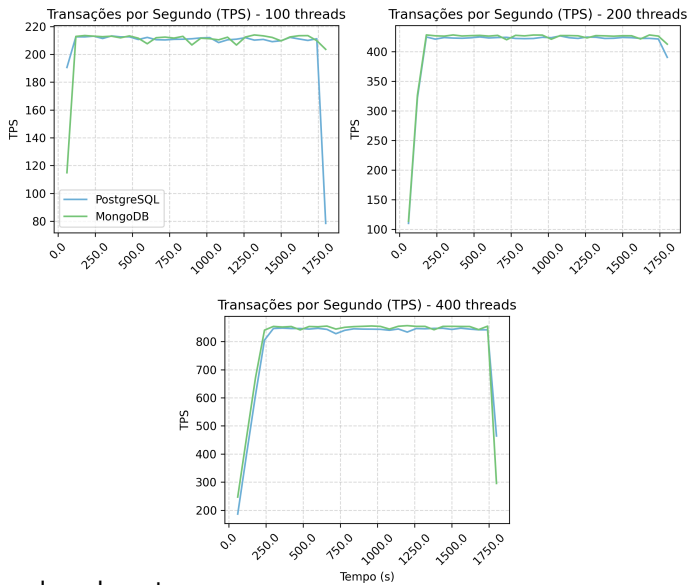
Fonte: Elaborado pelo autor.

A Figura 3 apresenta a taxa de transações por segundo (TPS) ao longo do tempo durante a execução da carga de inserção, nos cenários de 100, 200 e 400 threads. Em todos os níveis de concorrência, tanto o PostgreSQL quanto o MongoDB atingem rapidamente um platô de desempenho estável após a fase inicial de aquecimento.

Com 100 threads, ambos os bancos alcançam aproximadamente 210 TPS, com pequenas oscilações que se mantêm estáveis durante a maior parte do teste. A partir de 200 threads, os dois sistemas demonstram crescimento proporcional, mantendo uma taxa próxima a 420 TPS. No cenário com 400 threads, essa taxa dobra novamente, alcançando mais de 800 TPS, o que demonstra a capacidade de ambos os bancos de escalar com eficiência sob carga intensiva de inserções.

É notável que MongoDB e PostgreSQL apresentaram comportamentos quase idênticos ao longo dos testes, com pequenas variações momentâneas. A similaridade no desempenho sugere que, para workloads de escrita sequencial e sem operações complexas de consistência, ambos os bancos são igualmente eficazes. Esse padrão está de acordo com estudos como o de Khan et al. (2023), que destacam a capacidade dos bancos NoSQL de escalar horizontalmente, sem prejuízo no throughput, ao passo que bancos relacionais modernos, como o PostgreSQL, conseguem resultados comparáveis desde que bem configurados.

Figura 3 - Transações por segundo.

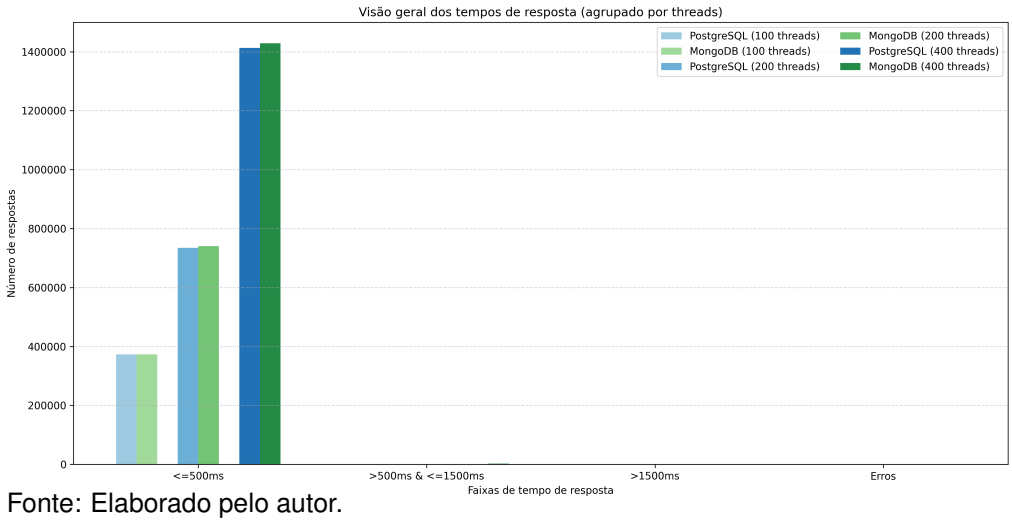


Fonte: Elaborado pelo autor.

A Figura 4 mostra que mais de 99% das requisições foram aten-

didas em até 500 ms, mesmo com o aumento da carga. Considerando apenas essa faixa, o PostgreSQL teve leve vantagem com 100 threads, mas o MongoDB superou em volume de respostas rápidas nos cenários com 200 e 400 threads, indicando maior estabilidade sob cargas mais altas.

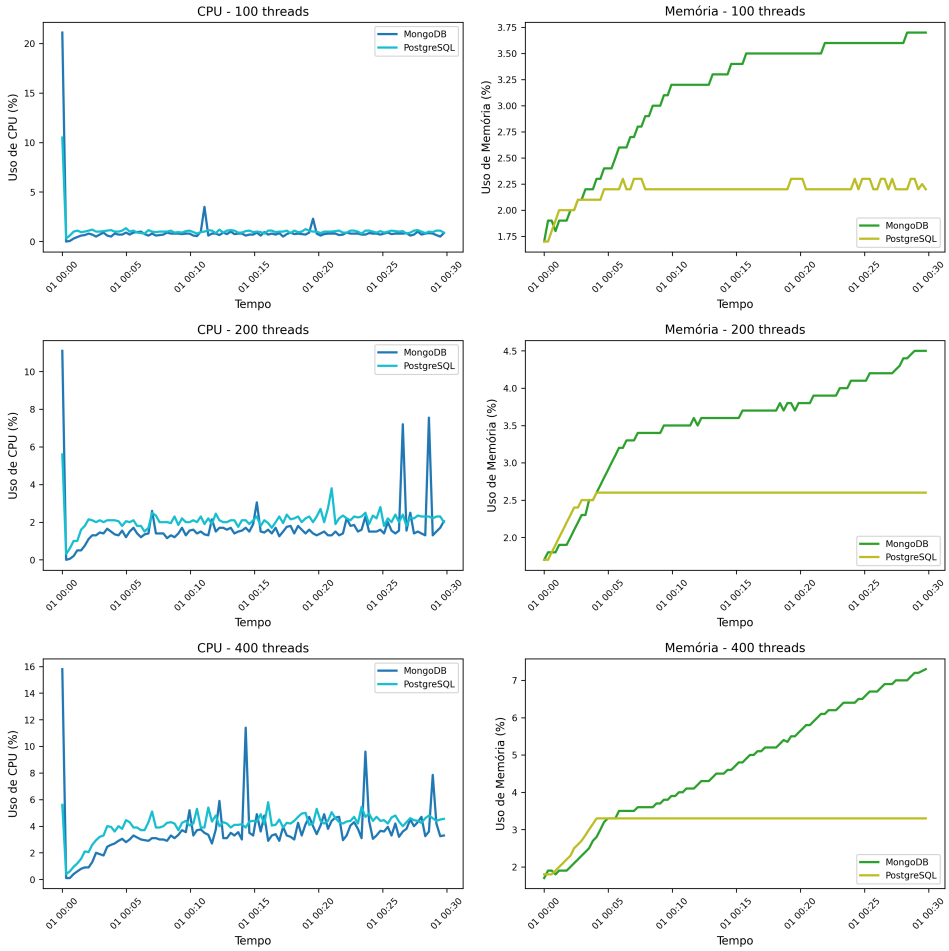
Figura 4 - Visão geral do tempo de resposta.



Fonte: Elaborado pelo autor.

A Figura 5 apresenta o comparativo de uso de CPU e memória pelos bancos PostgreSQL e MongoDB durante a execução da carga de inserção, com níveis crescentes de concorrência: 100, 200 e 400 threads. Nos três cenários, o PostgreSQL demonstrou um comportamento mais estável e eficiente, com uso de CPU oscilando pouco e memória praticamente constante em torno de 2,2% a 3,3%. Em contraste, o MongoDB apresentou crescimento contínuo no uso de memória conforme o número de threads aumentou, atingindo cerca de 7,2% no cenário com 400 threads. Além disso, seu uso de CPU mostrou variações pontuais mais bruscas, indicando maior sensibilidade à carga. Esses resultados sugerem que o PostgreSQL gerencia melhor os recursos do sistema em operações de escrita intensiva sob alta concorrência.

Figura 5 - Uso de recursos computacionais



Fonte: Elaborado pelo autor.

3.2 CARGA DE LEITURA (*READ-HEAVY WORKLOAD*)

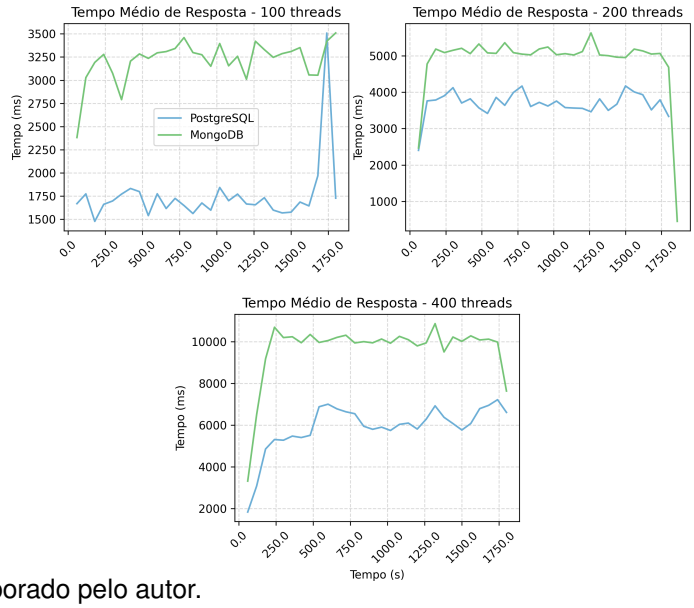
A Figura 6 apresenta o tempo médio de resposta ao longo da execução para 100, 200 e 400 threads no cenário de leitura intensiva. Com 100 threads, o MongoDB apresentou tempos médios consistentemente mais altos que o PostgreSQL, variando entre 3000 e 3500 ms, enquanto o PostgreSQL manteve-se abaixo de 1800 ms ao longo de toda a execução. Esse comportamento inicial já sugere maior eficiência do PostgreSQL na gestão de leituras concorrentes.

Com o aumento para 200 threads, o padrão se mantém. O MongoDB apresentou médias próximas ou acima de 5000 ms, ao passo que o PostgreSQL oscilou entre 3500 e 4000 ms, mantendo-se mais estável.

No cenário com 400 threads, a diferença se acentua: o PostgreSQL registrou tempos médios em torno de 6000 ms, enquanto o MongoDB superou os 10000 ms em diversos momentos da execução. Esses resultados indicam que o MongoDB sofre maior degradação de desempe-

nho sob alta concorrência em operações predominantemente de leitura, ao passo que o PostgreSQL demonstrou maior resiliência à carga.

Figura 6 - Tempo médio de resposta.



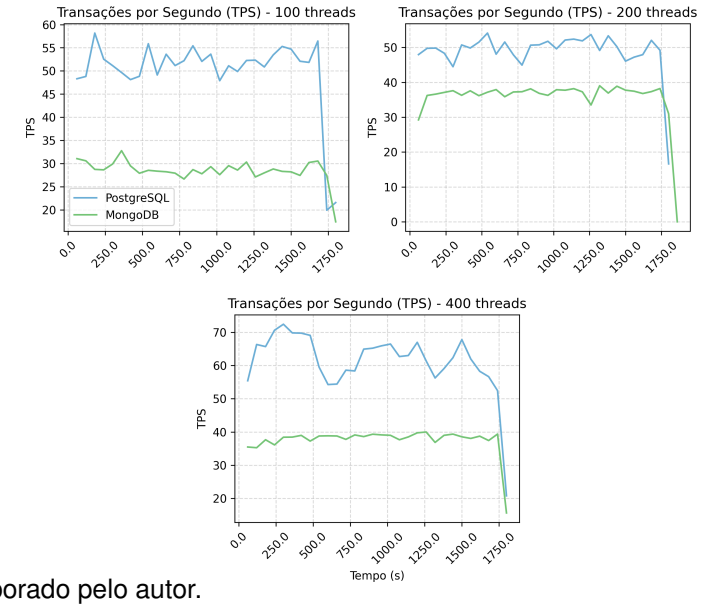
Fonte: Elaborado pelo autor.

A Figura 7 mostra o número de transações por segundo (TPS) ao longo do tempo para os testes com 100, 200 e 400 threads no cenário de leitura intensiva. Em todos os níveis de concorrência, o PostgreSQL superou o MongoDB em desempenho, mantendo taxas de transações mais elevadas e estáveis.

Com 100 threads, o PostgreSQL sustentou valores entre 48 e 58 TPS, enquanto o MongoDB oscilou em torno de 28 a 32 TPS. Esse padrão se repetiu com 200 threads, em que o PostgreSQL atingiu até 55 TPS, contrastando com os cerca de 36 TPS do MongoDB. O distanciamento se tornou ainda mais evidente com 400 threads, onde o PostgreSQL operou entre 55 e 70 TPS, e o MongoDB manteve-se em uma faixa inferior, entre 35 e 40 TPS.

Esse desempenho reforça a capacidade do PostgreSQL de escalar em leitura mesmo sob alta concorrência, um comportamento alinhado ao que já foi evidenciado por Khan et al. (2023) e Makris et al. (2021), que destacam o domínio do modelo relacional em consultas complexas e estruturadas.

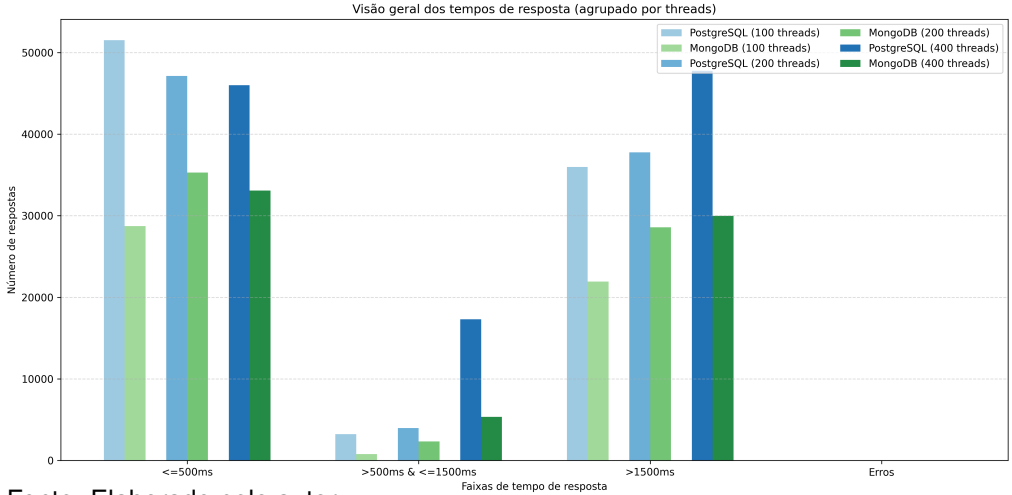
Figura 7 - Transações por segundo (TPS).



Fonte: Elaborado pelo autor.

A Figura 8 indica que o PostgreSQL concentrou mais respostas abaixo de 500 ms em todos os cenários. Com 100 threads, ultrapassou 50.000 respostas rápidas, enquanto o MongoDB ficou abaixo de 30.000. À medida que a carga aumentou, o MongoDB teve mais respostas acima de 1500 ms, mostrando menor consistência sob leitura intensiva.

Figura 8 - Visão geral do tempo de resposta.



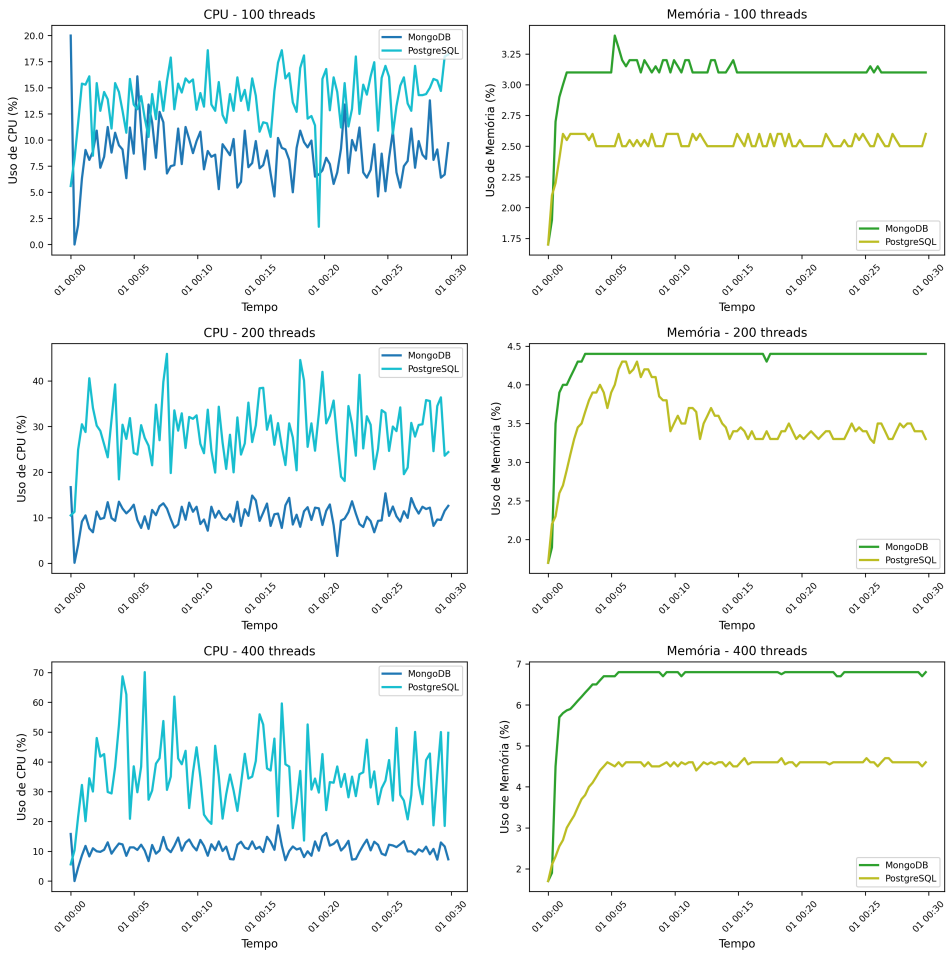
Fonte: Elaborado pelo autor.

A Figura 9 apresenta o comparativo de uso de CPU e memória pelos bancos PostgreSQL e MongoDB durante a execução da carga de leitura, com níveis crescentes de concorrência: 100, 200 e 400 threads. Em todos os cenários, o PostgreSQL apresentou maior uso de CPU em

relação ao MongoDB, especialmente com 400 threads, onde a utilização ultrapassou 60%, contra cerca de 20% do MongoDB.

Esse padrão indica que o PostgreSQL adota uma abordagem mais agressiva para explorar os recursos computacionais disponíveis em operações de leitura. Quanto ao uso de memória, o MongoDB teve um crescimento mais acentuado e rápido, estabilizando-se acima de 4% nas cargas mais altas, enquanto o PostgreSQL se manteve constante, próximo de 3%. Esses resultados sugerem que, para cargas predominantemente de leitura, o PostgreSQL utiliza mais CPU para manter desempenho elevado, enquanto o MongoDB depende mais fortemente de estratégias baseadas em cache de memória.

Figura 9 - Uso de recursos computacionais



Fonte: Elaborado pelo autor.

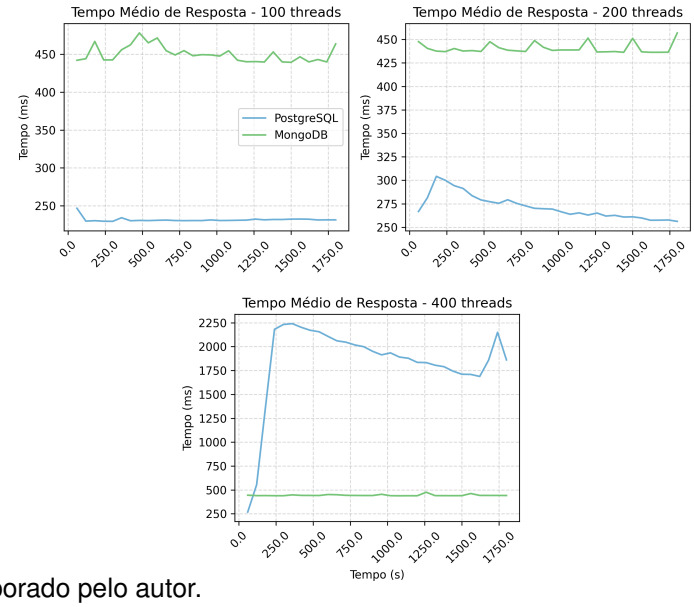
3.3 CARGA DE ATUALIZAÇÃO (*UPDATE-HEAVY WORKLOAD*)

A Figura 10 apresenta o tempo médio de resposta ao longo da execução para os cenários de 100, 200 e 400 threads na carga de atualização. Com 100 e 200 threads, o PostgreSQL apresentou desempenho

superior ao MongoDB, com tempos médios de resposta entre 230 e 300 ms, enquanto o MongoDB manteve-se acima de 440 ms em ambos os casos. No entanto, com 400 threads, o PostgreSQL sofreu degradação significativa de desempenho, ultrapassando os 2200 ms em diversos momentos, ao passo que o MongoDB manteve-se praticamente estável, com médias semelhantes às registradas nos testes anteriores.

Essa queda de desempenho do PostgreSQL pode ser atribuída à sua arquitetura transacional baseada em bloqueios e controle de concorrência mais rígido. Como discutido por Florencio et al. (2017), bancos relacionais tendem a sofrer sob cargas de escrita concorrente elevada devido à contenção de recursos e ao overhead de consistência. Já o MongoDB, com seu modelo mais flexível e operações menos dependentes de integridade relacional, demonstrou maior robustez nesse cenário mais extremo.

Figura 10 - Tempo médio de resposta.



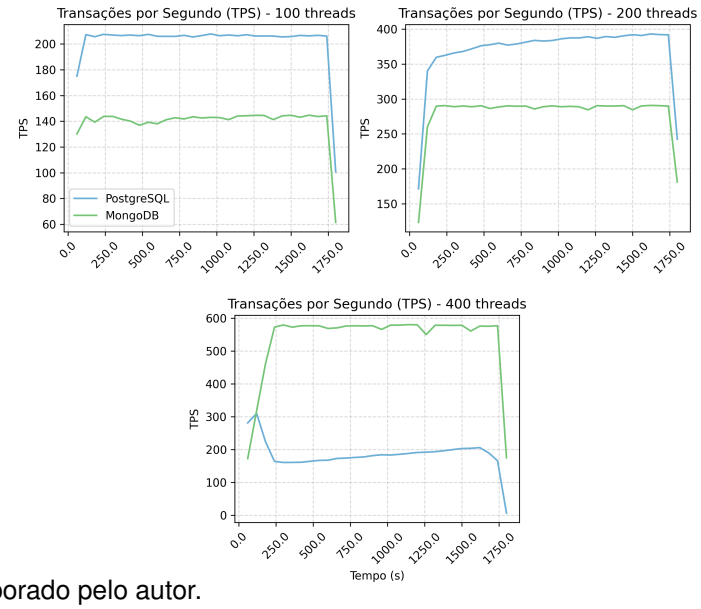
Fonte: Elaborado pelo autor.

A Figura 11 apresenta a evolução das transações por segundo (TPS) ao longo do tempo nos cenários de 100, 200 e 400 threads para a carga de atualização. Nos dois primeiros cenários (100 e 200 threads), o PostgreSQL demonstrou desempenho superior em termos de TPS, mantendo taxas estáveis acima de 200 e 380 transações por segundo, respectivamente, enquanto o MongoDB permaneceu consistentemente abaixo, com cerca de 140 a 290 TPS. Esse comportamento reforça a eficiência do PostgreSQL sob cargas moderadas de escrita.

No entanto, com o aumento da concorrência para 400 threads,

o cenário se inverte. O PostgreSQL apresentou uma queda significativa na taxa de transações, estabilizando-se em torno de 200 TPS, enquanto o MongoDB manteve sua escalabilidade, atingindo e sustentando cerca de 570 TPS ao longo da execução. Esse resultado confirma o padrão observado também no tempo médio de resposta: o PostgreSQL sofre degradação sob concorrência extrema, possivelmente devido à sua maior rigidez transacional e à contenção de recursos internos. O MongoDB, por sua vez, mostrou maior escalabilidade e capacidade de absorver altas taxas de atualização simultânea com estabilidade.

Figura 11 - Transações por segundo.

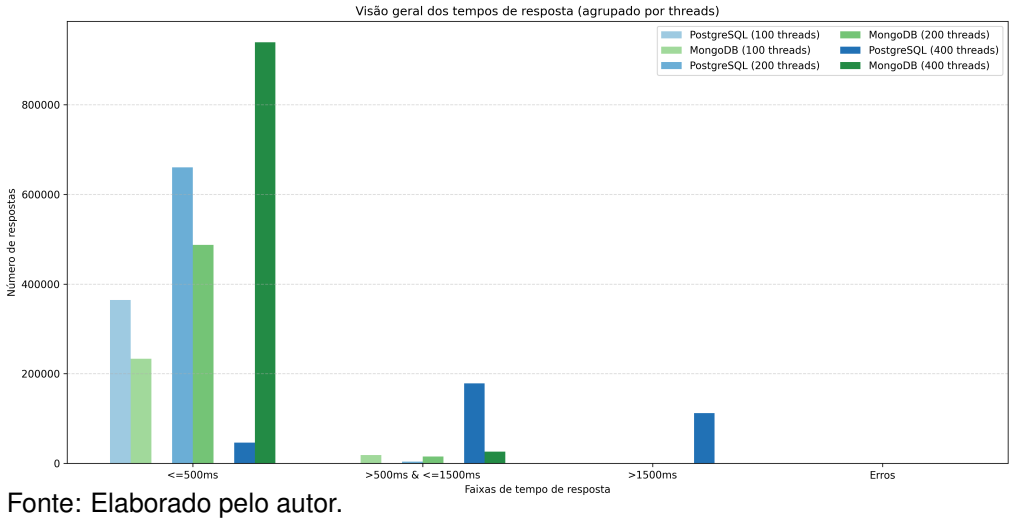


Fonte: Elaborado pelo autor.

A Figura 12 mostra que, com 100 e 200 threads, ambos os bancos atenderam a maioria das requisições em até 500 ms, com leve vantagem para o PostgreSQL. Em 400 threads, o MongoDB superou com mais de 900 mil respostas rápidas, enquanto o PostgreSQL concentrou-se nas faixas mais lentas. Isso indica maior resiliência do MongoDB sob escrita concorrente.

A ausência de erros em ambos os bancos nos três níveis de carga demonstra a robustez das soluções, mesmo sob cenários extremos de atualização concorrente. Ainda assim, o MongoDB demonstrou maior resiliência nesse tipo de workload, o que está em sintonia com os princípios arquiteturais apontados por Khan et al. (2023), que ressaltam o bom desempenho de bancos NoSQL em operações de escrita paralela e distribuída.

Figura 12 - Visão geral do tempo de resposta.

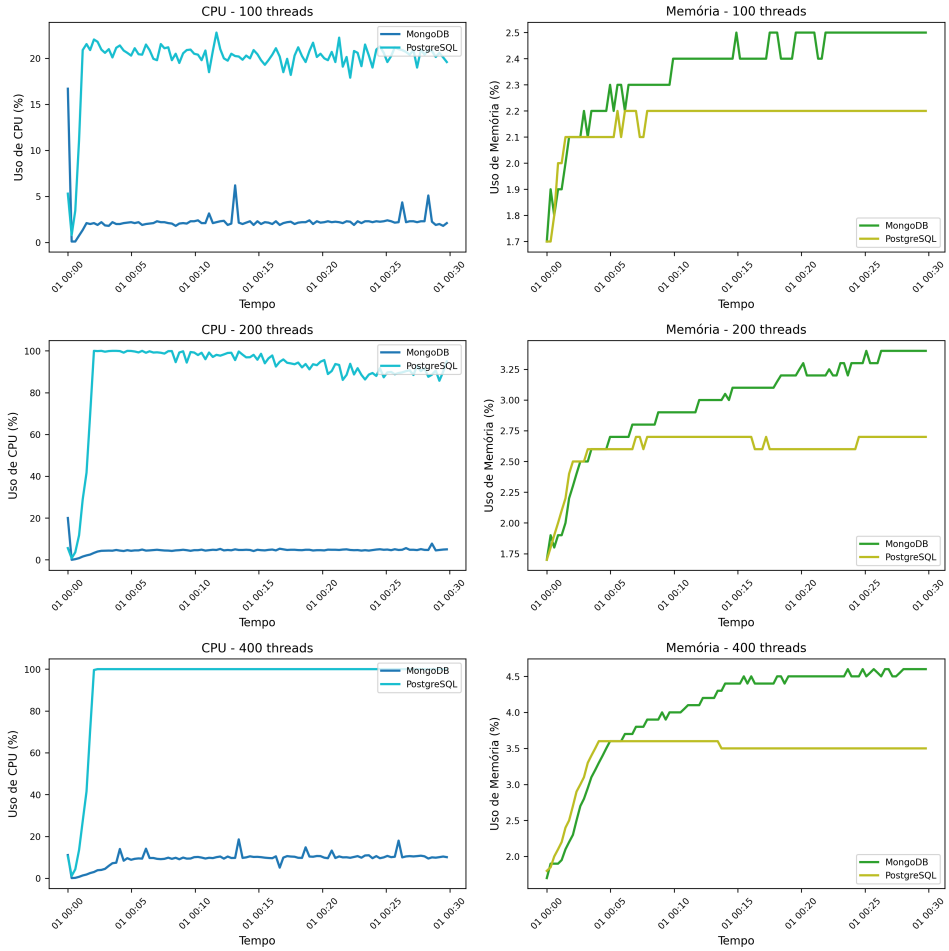


Fonte: Elaborado pelo autor.

A Figura 13 apresenta o comparativo de uso de CPU e memória pelos bancos PostgreSQL e MongoDB durante a execução da carga de atualização, com níveis crescentes de concorrência: 100, 200 e 400 threads. O PostgreSQL rapidamente atingiu o pico de uso de CPU, alcançando 100% já a partir de 200 threads e mantendo-se nesse patamar em 400 threads, demonstrando sua disposição em utilizar toda a capacidade computacional disponível para lidar com operações de escrita concorrente.

O MongoDB, por outro lado, apresentou uso de CPU significativamente mais baixo e estável ao longo dos testes. Em relação ao uso de memória, o MongoDB mostrou crescimento contínuo, passando de cerca de 1,9% para mais de 4,5% entre os cenários com 100 e 400 threads. Já o PostgreSQL manteve seu uso de memória praticamente fixo em torno de 3,2%. Esse contraste indica que o PostgreSQL tende a priorizar desempenho imediato via uso de CPU, enquanto o MongoDB acumula recursos em memória, o que pode afetar a previsibilidade sob alta concorrência de atualizações.

Figura 13 - Uso de recursos computacionais



Fonte: Elaborado pelo autor.

3.4 CARGA MISTA (*MIXED WORKLOAD*)

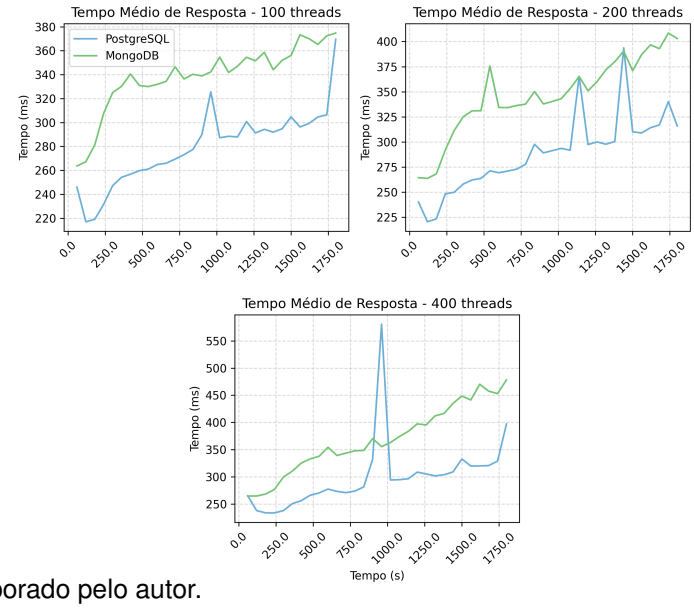
A Figura 14 apresenta o tempo médio de resposta ao longo do tempo para a carga mista, nos cenários com 100, 200 e 400 threads. Em todos os níveis de concorrência, o PostgreSQL obteve tempos de resposta inferiores aos do MongoDB, com menor variação e crescimento mais controlado ao longo da execução.

Com 100 threads, ambos os bancos iniciaram com tempos relativamente próximos, mas o MongoDB apresentou uma elevação mais acentuada, atingindo cerca de 370 ms, enquanto o PostgreSQL oscilou em torno de 300 ms. À medida que a concorrência aumentou para 200 threads, a diferença se manteve: o MongoDB ultrapassou os 400 ms, enquanto o PostgreSQL variou entre 250 e 380 ms, ainda apresentando desempenho mais favorável.

No cenário mais extremo, com 400 threads, a disparidade se acentuou. O MongoDB teve crescimento constante no tempo médio, atin-

gindo picos próximos a 470 ms. Já o PostgreSQL, embora com algumas oscilações, permaneceu abaixo desse limite durante a maior parte da execução. Esses resultados sugerem que, em cargas mistas com alta concorrência, o PostgreSQL mantém uma vantagem em termos de latência, provavelmente devido à sua arquitetura transacional e otimizações para operações simultâneas de leitura e escrita.

Figura 14 - Tempo médio de resposta.



Fonte: Elaborado pelo autor.

A Figura 15 apresenta a taxa de transações por segundo (TPS) para os bancos PostgreSQL e MongoDB sob carga mista, nos cenários com 100, 200 e 400 threads. Em todos os níveis de concorrência, o PostgreSQL manteve uma taxa de transações superior em relação ao MongoDB, com desempenho mais estável ao longo do tempo.

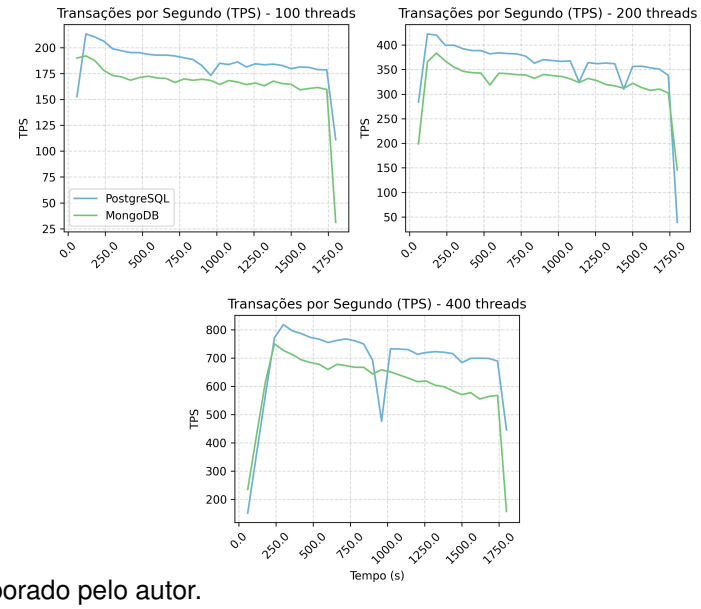
Com 100 threads, o PostgreSQL alcançou uma média acima de 200 TPS durante boa parte da execução, enquanto o MongoDB oscilou em torno de 180 TPS. Essa diferença já indica maior eficiência do PostgreSQL em lidar com a simultaneidade entre leitura e escrita.

No cenário com 200 threads, ambos os bancos demonstraram aumento nas taxas, mas o PostgreSQL permaneceu à frente, sustentando valores próximos de 400 TPS, contra cerca de 350 TPS do MongoDB. Essa superioridade se manteve mesmo com o aumento da carga.

Com 400 threads, o PostgreSQL obteve os melhores resultados, chegando a picos de aproximadamente 800 TPS, enquanto o MongoDB atingiu um patamar inferior, com média em torno de 650 TPS. Esses re-

sultados reforçam a capacidade do PostgreSQL em explorar o aumento da concorrência com maior aproveitamento dos recursos do sistema, mesmo em workloads balanceados entre leitura e escrita. Isso evidencia sua robustez em cenários com operações simultâneas e alta demanda transacional.

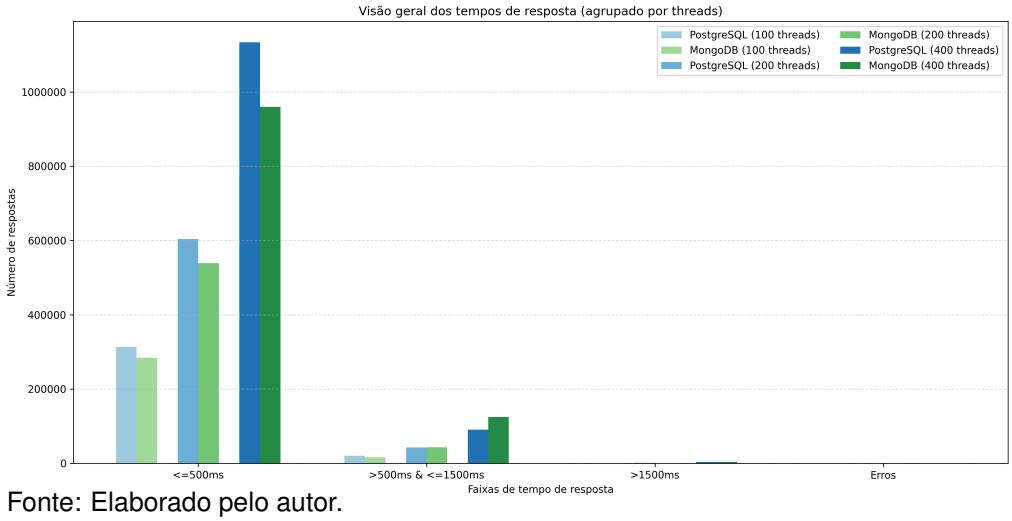
Figura 15 - Transações por segundo.



Fonte: Elaborado pelo autor.

A Figura 16 mostra que a maioria das requisições foi concluída em até 500 ms, com destaque para o PostgreSQL em 400 threads. O MongoDB também teve bom desempenho nessa faixa, mas apresentou mais requisições intermediárias (>500 ms) sob alta carga. Em geral, ambos mantiveram desempenho estável, sem erros.

Figura 16 - Visão geral do tempo de resposta.

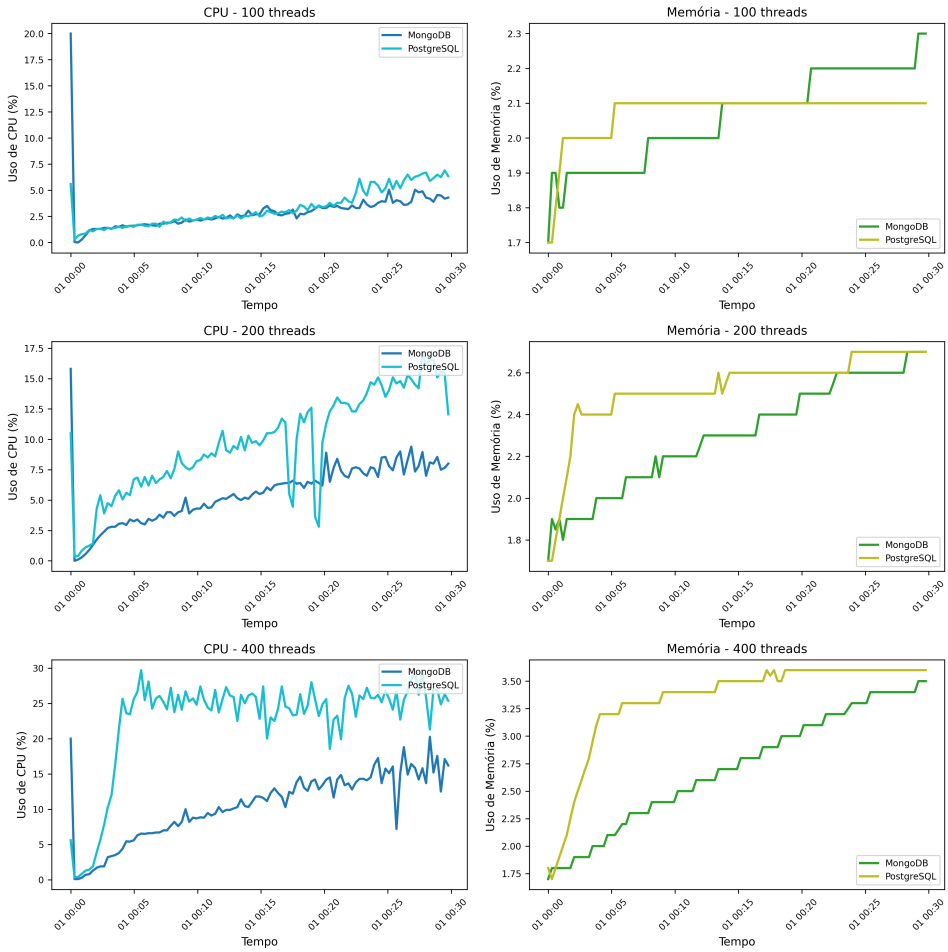


Fonte: Elaborado pelo autor.

A Figura 17 apresenta o comparativo de uso de CPU e memória pelos bancos PostgreSQL e MongoDB durante a execução da carga mista, com níveis crescentes de concorrência: 100, 200 e 400 threads. O PostgreSQL apresentou aumento progressivo no uso de CPU conforme a carga aumentou, ultrapassando 25% com 400 threads.

O MongoDB, embora com uso mais contido, também apresentou elevação gradual, mas sem picos expressivos. No que diz respeito ao uso de memória, o MongoDB voltou a apresentar crescimento constante, chegando a aproximadamente 3,7% no maior nível de concorrência, enquanto o PostgreSQL manteve seu consumo estável próximo a 2,3%. Esses resultados indicam que, mesmo em cenários híbridos, o PostgreSQL sustenta um gerenciamento mais previsível dos recursos, enquanto o MongoDB adota estratégias mais dinâmicas, especialmente na alocação de memória.

Figura 17 - Uso de recursos computacionais



Fonte: Elaborado pelo autor.

4 CONCLUSÃO

Este trabalho comparou o desempenho dos bancos PostgreSQL e MongoDB sob diferentes cargas de trabalho e níveis de concorrência, utilizando técnicas de stress e load testing em cenários realistas. A análise evidenciou que o PostgreSQL é superior em operações de leitura e cargas mistas moderadas, demonstrando maior estabilidade e uso eficiente da CPU. Por outro lado, o MongoDB apresentou desempenho mais robusto em cenários de escrita intensiva e alta concorrência, com melhor escalabilidade e aproveitamento de memória.

Esses resultados indicam que, embora ambos os bancos sejam tecnicamente viáveis, o PostgreSQL se destaca em aplicações com forte demanda de leitura estruturada, enquanto o MongoDB é mais indicado para aplicações com alta inserção concorrente e esquemas flexíveis. Portanto, a escolha do banco de dados não deve ser neutra: ela deve refletir o perfil predominante de carga da aplicação.

Para trabalhos futuros, propõe-se não apenas expandir os testes para ambientes distribuídos e incluir bancos emergentes, mas também investigar o impacto de otimizações específicas — como uso de cache externo, replicação e particionamento horizontal — na performance dos sistemas. Avaliar se essas técnicas alteram significativamente o comportamento de cada banco poderá oferecer recomendações ainda mais precisas para aplicações em ambientes de alta demanda.

REFERÊNCIAS

CORBELLINI, A. et al. **Persisting big-data: The NoSQL landscape**. [S.l.]: Information Systems, 2017. v. 63. 1-23 p.

COSTA, M. et al. **Database encryption performance impact on PostgreSQL and MongoDB**. [S.l.]: Springer, 2022. 121–127 p.

FIGUEIREDO, D. et al. **Performance evaluation between HarperDB, MongoDB and PostgreSQL**. [S.l.]: Springer, 2022. 85–94 p.

FLORENCIO, D. A. et al. **Which Fits Better? A Comparative Analysis about NoSQL Key-Value Databases**. [S.l.]: IEEE Latin America Transactions, 2017. v. 15. 2251–2256 p.

HASSAN, M. A. **Relational and NoSQL databases: the appropriate database model choice**. Muscat, Omã: IEEE, 2021. 1-6 p.

KHAN, W. et al. **SQL and NoSQL Database Software Architecture Performance Analysis and Assessments—A Systematic Literature**

Review. [S.l.]: Big Data and Cognitive Computing, 2023. v. 7. ISSN 2504-2289.

MAKRIS, A. et al. **MongoDB vs PostgreSQL: A comparative study on performance aspects**. [S.l.: s.n.], 2021. v. 25. 243–268 p.

MICROSOFT. **O que é um banco de dados NoSQL?** 2025. Acessado em: 18 de março de 2025. Disponível em: <https://azure.microsoft.com/pt-br/resources/cloud-computing-dictionary/what-is-nosql-database>.

NURIEV, M. et al. **Enhancing MongoDB query performance through index optimization**. [s.n.], 2024. v. 531. 03022 p. Disponível em: <https://doi.org/10.1051/e3sconf/202453103022>.

ORACLE. **O que é um banco de dados relacional (RDBMS)?** 2025. Acessado em: 16 de março de 2025. Disponível em: <https://www.oracle.com/br/database/what-is-a-relational-database/>.

PARGAONKAR, S. **A comprehensive review of performance testing methodologies and best practices: Software quality engineering**. International Journal of Science and Research (IJSR), 2023, v. 12, p. 2008–2014.

PRADEEP, S.; SHARMA, D. Y. **A pragmatic evaluation of stress and performance testing technologies for web based applications**. 2019, p. 399–403.

SILVA, L. F. da; LIMA, J. V. F. **An evaluation of relational and NoSQL distributed databases on a low-power cluster**. [S.l.]: The Journal of Supercomputing, 2023.

YEDILKHAN, D. et al. **Performance analysis of scaling NoSQL vs SQL: a comparative study of MongoDB, Cassandra, and PostgreSQL**. Astana, Cazaquistão: IEEE, 2023. 479-483 p.

YENUGULA, M.; KODAM, R.; HE, D. **Performance and load testing: Tools and challenges**. International Journal of Engineering in Computer Science, 2019, v. 1, p. 57–62.