Universidade de Brasília
Faculdade de Tecnologia

Análise do tempo de resposta em Bancos de Dados Não-Relacionais (NoSQL) usados para armazenamento de imagens

Caio Alessandri Albuquerque

Lucas Monteiro Miranda

PROJETO FINAL DE CURSO

ENGENHARIA DE CONTROLE E AUTOMAÇÃO

Brasília

2023

Universidade de Brasília
Faculdade de Tecnologia

Análise do tempo de resposta em Bancos de Dados Não-Relacionais (NoSQL) usados para armazenamento de imagens

Caio Alessandri Albuquerque
Lucas Monteiro Miranda

Projeto Final de Curso submetido como requisito parcial para obtenção do grau de Engenheiro de Controle e Automação

Orientador: Prof. Dr. Flávio de Barros Vidal

Brasília
2023

Alessandri Albuquerque, Caio.
A372a Análise do tempo de resposta em Bancos de Dados Não-Relacionais (NoSQL) usados para armazenamento de imagens / Caio Alessandri Albuquerque; Lucas Monteiro Miranda; orientador Flávio de Barros Vidal. -- Brasília, 2023.
114 p.

Projeto Final de Curso (Engenharia de Controle e Automação) -- Universidade de Brasília, 2023.

1. Banco de Dados. 2. Tempo de resposta. 3. Desempenho de Acesso. 4. NoSQL. I. Monteiro Miranda, Lucas. II. Barros Vidal, Flávio de, orient. III. Título

Universidade de Brasília
Faculdade de Tecnologia

Análise do tempo de resposta em Bancos de Dados Não-Relacionais (NoSQL) usados para armazenamento de imagens

Caio Alessandri Albuquerque
Lucas Monteiro Miranda

Projeto Final de Curso submetido como requisito parcial para obtenção do grau de Engenheiro de Controle e Automação

Trabalho aprovado. Brasília, 24 de Julho de 2023:

Prof. Dr. Flávio de Barros Vidal,
UnB/IE/CIC
Orientador

Prof. Dr. Marcos Fagundes Caetano,
UnB/IE/CIC
Examinador interno

Prof. Dr. Marcus Vinicius Lamar,
UnB/IE/CIC
Examinador interno

Brasília
2023

Dedico essa jornada aos três, e aos cinco. A Europa é um ótimo lugar nessa época do ano.

Caio Alessandri Albuquerque

Dedico esse trabalho à Fire, meu segundo lar.

Lucas Monteiro Miranda

Agradecimentos

Agradeço os esforços de minha família para me levar a essas oportunidades, por todos os lados.

Agradeço ao Prof. Dr. Flávio de Barros Vidal pela orientação, ajudas com o projeto, e preocupações além do acadêmico, sempre empenhado pelo justo.

Agradeço ao amigos que ganhei e perdi, que me permitiram continuar em frente e me lembrar de meus objetivos. Especialmente ao Lucas, por sua infindável paciência e confiança ao longo dos estudos.

Também e finalmente agradeço Yuki, Alexis, Nick, Jacob, Pedro e tantos outros que me permitiram chegar até aqui.

Caio Alessandri Albuquerque

Agradeço, primeiramente, à Deus, pela saúde e pela força de vontade que me concedeu para finalizar essa etapa da minha jornada acadêmica.

Agradeço à minha família pelo constante apoio, motivação e ensino que me deram para que eu pudesse chegar até aqui.

Agradeço ao Prof. Dr. Flávio de Barros Vidal pela excelente orientação nesse projeto, sempre disposto a tirar nossas dúvidas e nos ajudar a encontrar as melhores soluções.

Agradeço aos meus amigos pelos incontáveis momentos que compartilhamos, sejam eles de diversão ou de luta, e que me ajudaram a prosseguir minha jornada. Em especial, agradeço ao Caio, por ter aceitado embarcar nessa aventura comigo, e ao Delano, pois sem ele, eu não teria finalizado esse curso.

Por fim, agradeço à Íris, Crifina, Gaia, Réquiem e Planes, pelas aventuras fantásticas. A sorte sempre esteve ao meu lado, e mesmo em uma montanha escura com nada além de uma porta, uma simples invenção capaz de congelar um reino inteiro pode surgir.

Lucas Monteiro Miranda

“The creation of a single world
comes from a huge number of fragments and chaos.”
(Hayao Miyazaki)

Resumo

O armazenamento de imagens em bancos de dados é um extenso tópico de discussão, com diferentes técnicas sendo empregadas hoje no mercado. Com a diferença arquitetural entre bancos de dados SQL e NoSQL, é esperado que existam vantagens e desvantagens no uso de cada um destes. Sendo assim, esse trabalho tem por objetivo observar, medir e interpretar as diferenças de desempenho de diversos bancos de dados, baseados em SQL e NoSQL, na inserção, procura e armazenamento de diversas imagens para utilização em um sistema de processamento de imagens para coleta de lixo. Foram utilizadas métricas definidas igualmente para cada banco de dados, coletadas com o uso de um *script* automatizado desenvolvido para esse propósito, com a menor interferência possível e bibliotecas especializadas para a medição do tempo de execução. Foi observado que a utilização do modelo NoSQL é mais propícia para o caso de uso, com opções como o Redis apresentando eficiência duas ordens de magnitude maiores em alguns dos testes.

Palavras-chave: Banco de Dados. Tempo de resposta. Desempenho de Acesso. NoSQL.

Abstract

Storing images in databases is an extensive topic of discussion, with different techniques being used in today's market. With the architectural differences between SQL and NoSQL databases, it is expected to have advantages and disadvantages when using one or the other. With that being said, this paper's objective is to observe, measure and interpret the differences in performance of various databases, both SQL and NoSQL based, when working with insertion, search, and general storage of numerous images in a database that will be used for an image processing system for garbage collecting. Equally defined metrics were used for each database, measured with a Python script developed for that purpose. It was possible to observe that the usage of NoSQL models is more conducive to the use case, with options such as Redis showing efficiency two orders of magnitude greater in some of the tests.

Keywords: Database. Response time. Access performance. NoSQL.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

ADO	*ActiveX Data Objects* . . . 31
API	*Application Programming Interface* . . . 34
AWS	*Amazon Web Services* . . . 38
BD	Banco de Dados . . . 41
BLOB	*Binary Large Object* . . . 18
BSON	*Binary JSON* . . . 33
CPU	*Central Processing Unit* . . . 24
CQL	*Cassandra Query Language* . . . 32
CSV	*Comma Separated Values* . . . 34
D	Deleção . . . 73
GPL	*General Public License* . . . 31
HTTP	*Hypertext Transfer Protocol Markup Language* . . . 34
IR	Inserção em rajada . . . 73
IS	Inserção única . . . 73
ISAM	*Indexed Sequential Access Method* . . . 22
JSON	*JavaScript Object Notation* . . . 23
L	Leitura . . . 73
Mem	Memória . . . 38
NoSQL	*Not Only Structured Query Language* . . . 18
ODBC	*Open Database Connectivity* . . . 31
PHP	*PHP: Hypertext Preprocessor* . . . 31
RDBMS	*Relational Database Management System* . . . 21
Redis	*Remote Dictionary Server* . . . 27
RESTful	*Representational State Transfer* . . . 34
Scala	*Scalable Language* . . . 32
SQL	*Structured Query Language* . . . 18
SSD	*Solid State Drive* . . . 33
SSPL	*Server Side Public License* . . . 32
UnB	Universidade de Brasília . . . 38
vCPU	*Virtual Central Processing Unit* . . . 38
XML	*Extensible Markup Language* . . . 34

Sumário

1 Introdução

1.1 Contextualização

A necessidade de utilização de bancos de dados para armazenar imagens aparece diretamente ligada à utilização de imagens em processamentos computacionais, sendo necessária a catalogação, referência e relação entre as imagens para o bom funcionamento de algoritmos, de forma especial quando se trata de redes neurais. Nessa consideração, temos diferentes tipos de bancos de dados (principalmente, SQL e NoSQL) para armezenar essas imagens. Os modelos SQL são conhecidos por serem custosos para realizar o armazenamento e acesso de BLOBs (arquivos binários que representam as imagens), tendo nos NoSQLs soluções mais flexíveis, que podem ser exploradas no contexto de arquivação de imagens.

Com poucos estudos concretos na avaliação desses bancos de dados para a utilização com imagens em larga escala, temos uma situação em que diferentes bancos são usados na suposição de serem otimizados, o que mostra necessário dados mais concretos.

1.2 Justificativa

É de comum conhecimento que arquivos binários não devem ser armazenados diretamente em um banco de dados relacional, costumeiramente sendo gravados em disco e armazenando no banco de dados apenas um ponteiro para o local, como descrito por Elmasri e Navathe (2016, p. 560–561). Porém, Sebastian e Aelterman (2012, p. 23–24) apontam alguns perigos da utilização desse método, como as imagens não serem incluída nos *backups* de emergência do banco de dados ou a possibilidade de que sejam alteradas por terceiros, assim ficando fora de sincronia com o banco de dados.

Sebastian e Aelterman (2012, p. 24–30) trazem a ideia de que o modelo relacional possui certos requisitos para uma boa operação, e que dados estruturados atendem a esses requisitos e portanto são adequados para o modelo. Porém, imagens não são dados estruturados. Uma imagem é o que é conhecido como BLOB (do inglês, *Binary Large Object* - Arquivo Binário Largo), que de acordo com a Mozilla (2023), é um objeto semelhante a um arquivo, geralmente longo, que possui dados imutáveis.

Surge então a pergunta: "por que não utilizar bancos de dados não-relacionais?". De acordo com Harrison (2016, p. 51, 148), modelos não-relacionais, tal qual o modelo chave-valor, não apresentam uma definição tão restrita quanto os relacionais sobre o que pode ser armazenado, o que torna bancos de dados NoSQL bons candidatos ao uso para armazenamento de imagens.

Dessa forma, este trabalho foi desenvolvido para responder os questionamentos acerca da viabilidade do uso de bancos de dados não-relacionais para o armazenamento de imagens, quão grande é a diferença de seus desempenhos em relação aos relacionais, e por fim, dentre alguns exemplares dentre as várias tecnologias disponíveis no mercado, qual deles é o melhor para cada um dos cenários propostos. Desse modo buscamos a otimização interna de cada opção, sem a interferência da velocidade de acesso em rede remota.

1.3 Objetivos

Para o desenvolvimento desse trabalho, foram identificados os seguintes objetivos:

- Entender as diferenças no processo de armazenamento de imagens em bancos de dados SQL e NoSQL;
- Estimar medidas, de maneira padronizada, dos bancos de dados em estudo no trabalho;
- Obter, através da análise de resultados, os melhores casos de cada tipo de banco de dados para utilização no sistema em questão.

O primeiro ponto permitirá a formulação de hipóteses acerca de quais são as vantagens e desvantagens do uso de cada arquitetura de banco de dados para armazenamento de imagens, nos permitindo ter um panorama geral sobre os caminhos de implementação mais adequados.

O segundo objetivo trará dados confiáveis para a análise dos bancos de dados, servindo como confirmação ou invalidação das hipóteses obtidas na etapa anterior.

Por fim, o terceiro ponto será a reunião do conhecimento adquirido para gerar, de maneira concisa, referências acerca de quais bancos de dados se comportam mais idealmente à situação apresentada.

1.4 Organização do Trabalho

Este trabalho consiste dos seguintes capítulos:

- **Capítulo 2: Fundamentação Teórica**. Esse capítulo é responsável por explicar os fundamentos teóricos a respeito das tecnologias envolvidas neste trabalho. Serão discutidos conceitos a respeito de bancos de dados relacionais (SQL) e não-relacionais (NoSQL), bem como especificidades e métodos para a análise de desempenho destes.
- **Capítulo 3: Trabalhos Relacionados**. Nesse capítulo são apresentados outros trabalhos publicados a respeito do assunto de interesse deste estudo, falando de maneira breve sobre seus objetivos e resultados e como eles estão relacionados à nossa pesquisa.

- **Capítulo 4: Metodologia Proposta**. Aqui será apresentada a metodologia utilizada para a condução deste trabalho. Serão detalhados os métodos utilizados para a escolha dos bancos de dados estudados, quais métricas foram definidas para a análise, como essas métricas foram colocadas em teste e como os resultados foram coletados.
- **Capítulo 5: Resultados**. Este capítulo apresenta os resultados obtidos após os experimentos propostos nesse trabalho. Através da apresentação de gráficos para as métricas coletadas e tabelas comparativas, é feita uma discussão a respeito dos resultados, de maneira a definirmos quais bancos de dados se saíram melhor em cada um dos cenários propostos.
- **Capítulo 6: Conclusões**. O último capítulo deste trabalho apresenta uma visão geral sobre os resultados encontrados, bem como especula sobre possíveis melhorias a serem realizadas no futuro para dar continuidade ao assunto estudado.

2 Fundamentação Teórica

Este capítulo apresentará os principais conceitos teóricos envolvidos no desenvolvimento deste trabalho.

Uma apresentação básica sobre o conceito de banco de dados está fundamentada na necessidade de armazenamento de documentos e valores relacionados entre si. Sendo buscada a utilização em diversas escalas, esse armazenamento sofreu mudanças e otimizações em seu crescimento, principalmente no contexto de imensas quantidades de informações relacionadas sendo utilizadas em sistemas de algoritmos neurais e inteligência artificial, visto em Elmasri e Navathe (2016, p. 245).

Com o objetivo de melhor lidar com diferentes tipos de arquivos, e modos de relacionamentos de dados, alguns métodos de arquivação foram desenvolvidos.

2.1 Bancos de dados relacionais (SQL)

Os bancos de dados baseados em SQL (do inglês *Structured Query Language* - Linguagem de Consulta Estruturada) são formatados para terem relações de chave compartilhada entre tabelas, com colunas relacionando seus dados. Conjuntos similares de dados são colocados em tabelas a nível lógico, e os dados em si são armazenados em arquivos de página, cada um 8KiB em tamanho. Esses arquivos são estruturados de acordo com o RDBMS (do inglês *Relational Database Management System* - Sistema de Gerenciamento de Banco de Dados Relacional) específico sendo utilizado, como na Figura 2.1. Dependendo do RDBMS escolhido, a velocidade de acesso e gerenciamento pode variar, como dito em Harrington (2009, p. 9–10).

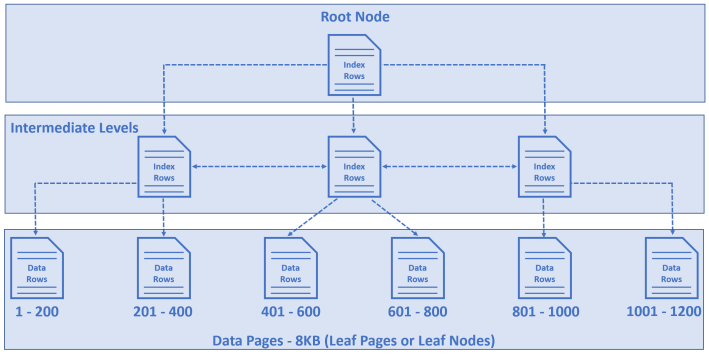

Figura 2.1 – Exemplo de armazenamento de Páginas de Dados em Árvore-B

Fonte:
https://www.pragimtech.com/blog/sql-optimization/how-is-data-stored-in-sql-database/Araujo (2015, p. 24)

A utilização do SQL suplantou o método mais comumente utilizado, ISAM (do inglês, *Indexed Sequential Access Method* - Método de Acesso Sequencial Indexado). Se mostrando uma alternativa mais segura, confiável e com melhor desempenho, mesmo que precisando de maior manuntenção em um sistema mais complexo, visto por Harrington (2009, p. 391–393). Definindo sua linguagem em volta de 4 modos de operação, Consulta de Dados, Manipulação de Dados, Definição de Dados e Controle de Acesso aos Dados, o SQL é bem definido e armazenado, uma opção rápida e concisa para operações básicas com informações. Mesmo com várias facilidades, ainda possui grandes desvantagens para o armazenamento de arquivos binários grandes. O servidor SQL precisa processar a leitura desses arquivos em todas as consultas, possivelmente diminuindo a eficiência no acesso dentro de cada tabela.

2.2 Bancos de dados não-relacionais (NoSQL)

No caso de bancos de dados NoSQL, temos ainda a ideia de relacionamento de dados, mas não apenas de modo tabular, como é o caso no padrão SQL. Nesse modo de organização, a estruturação do banco de dados pode ser feita por documentos, pares de chave-valor, armazenamento de registro extensível, grafos ou outros relacionamentos de informação que não são limitados ao tabular. A propriedade flexível dos formatos NoSQL permitem algumas otimizações, como maior escalonamento horizontal (usando também fragmentação ou *Sharding*, Figura 2.2, mostrado em Sadalage e Fowler (2013, p. 38–40)) e maior velocidade de acesso, já que é otimizado para entrega. Parte da consistência na entrega

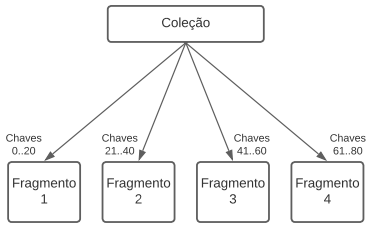

Figura 2.2 – Exemplo de fragmentação de dados em múltiplos servidores. Fonte: adaptado de Ken W. Alger (2017)

dos dados pode ser comprometida pela otimização de entrega. Com o tempo necessário para cada parte do banco de dados ser atualizada, tendo mais fragmentos, existe a possibilidade de registros estarem em suas versões antigas por breves períodos de tempo. Para bancos de dados que armazenam coleções de grande volume, diminuir o número de operações acaba ainda assim ser muito benéfico ao tempo necessário para cada operação, como pode ser o caso de armazenamento de arquivos binários, que precisam ser acessados por completo em modelos SQL para responder às operações requisitadas.

Os diferentes tipos de bancos NoSQL estudados tem definições próprias.

Chave	Valor
ID 001	Sessão A
ID 002	Sessão B

Tabela 2.1 – Exemplo de relação Chave-Valor

2.2.1 Chave-Valor

Estruturado em um modelo dicionário, sem precisar de um formato pré-definido, é uma opção para aumentar a performance, sendo mais leve e consumindo menos recursos. Uma ótima opção para cache, armazenar recomendações, gerenciar sessões de usuários, como em Sadalage e Fowler (2013, p. 20–21).

2.2.2 Documentos

No formato de Documentos, temos grupos de chave-valor armazenados em arquivos. A unidade básica de organização é o documento, que pode ser agrupado em coleções baseadas em funcionalidade, em Sadalage e Fowler (2013, p. 20–21). Possível armazenar dados sem definir uma *Schema*, transferindo o modelo diretamente ao documento. Como exemplo, temos o modelo JSON:

```
{
    "ID" : "001",
    "Nome" : "Eduardo",
    "Cargo" : "Zelador",
}
```

Sendo possível armazenar dados aninhados em sub-níveis. É um modelo flexível que permite desenvolvimento contínuo dos dados.

2.2.3 Série temporal

Séries temporais são coleções de dados compostas primariamente de três elementos: o momento de registro do dado, um identificador e o valor associado, como em Sadalage e Fowler (2013, p. 21–23). Esse método de armazenamento agrupa dados com identificadores e ordena com o tempo. É baseado em colunas, o que permite maior eficiência na recuperação dos dados e menor utilização de espaço em disco.

2.2.4 Grafos

O modelo de Grafos é composto por nós, que armazenam dados e bordas, que relacionam os nós. Ambos possuem propriedades definidas, facilitando a consulta no banco. Com as relações entre propriedades, é possível recuperar dados com apenas uma operação,

diminuindo os custo de acesso e permitindo a utilização das relações em operações posteriores. Ainda é flexível, adicionando nós e bordas sem precisar redefinir o sistema, Sadalage e Fowler (2013, p. 26–28).

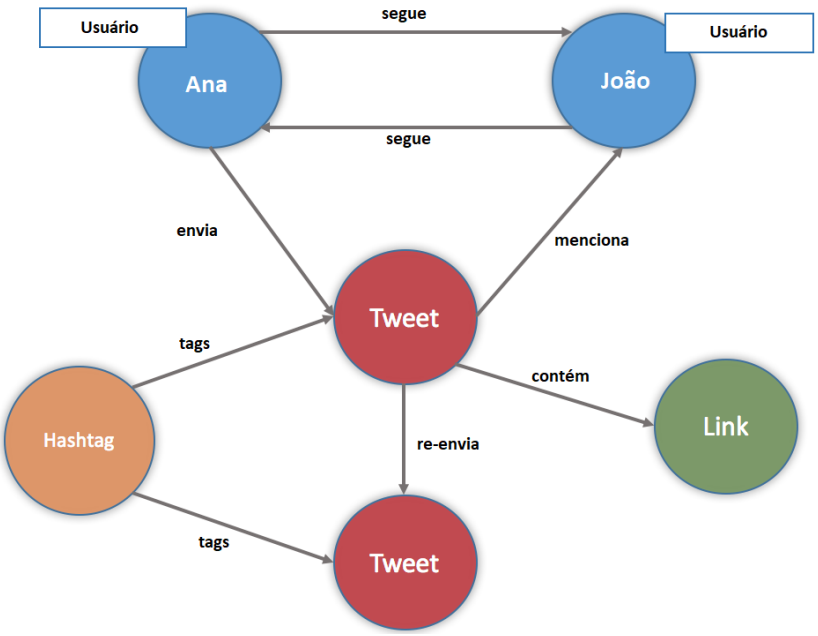

Figura 2.3 – Exemplo de relacionamento de Grafo. Fonte: adaptado de Dataversity (2016)

Com mais opções de armazenamento para demandas específicas, não estando limitados apenas ao armazenamento no padrão SQL, é possível achar modelos NoSQL que se encaixam ao problema explorado. Desse modo, vale também analisar qual das possibilidades resulta nas melhores métricas buscadas.

2.3 Análise de Desempenho

Tendo agora mais opções a serem exploradas para o armazenamento de valores é necessário metrificar o impacto que cada opção resulta no conjunto de dados escolhido. No caso SQL, mesmo com modos similares de armazenamento tabular, diferentes bancos de dados implementam suas próprias maneira de otimizar a entrega de dados, ou de facilitar algumas operações específicas, visto em Elmasri e Navathe (2016, p. 680–681). Para NoSQL, o próprio modo de armazenamento já muda o impacto de cada operação. No estudo, serão utilizados arquivos binário de tamanho variável (imagens), o que pode apresentar desafios ainda maiores para a escolha do banco. Com grandes sequências binárias é imperativo que o banco de dados seja capaz de buscar registros, incluir novos registros e deletar os que já não são mais necessários com eficiência.

Existem fatores que influenciam essa metrificação. No caso do próprio *hardware* da máquina, temos memórias cache que podem impactar a velocidade de acesso após suficientes operações semelhantes, instruções de operação na CPU que se otimizam para certos cálculos e a própria carga de trabalho da máquina no momento do acesso. Com o objetivo de metrificar todos os bancos de dados escolhidos para estudo de modo similar, é necessário definir as operações possíveis, métodos para mitigar as interferências externas e formatação dos resultados. Busca-se a otimização na interface com o conjunto de dados escolhido, então o tempo necessário para cada operação dentro do banco ser realizada é o ponto de comparação entre os bancos.

As operações que se relacionam com o armazenamento de imagens são primariamente (Elmasri e Navathe (2016, p. 304–328)):

- Inserção - Inserção inicial e posteriores novas imagens
- Leitura - Recuperação da imagem requisitada
- Atualização - Mudança no conteúdo de metadados relacionado
- Deleção - Remoção de registros não mais necessários

A análise de desempenho envolve melhor otimização com o menor número de problemas resultante. Isso significa a menor quantidade de erros nas operações, o consumo de recursos, o escalonamento dos dados e a facilidade de interação do banco com os gerenciadores.

3 Trabalhos Relacionados

O catálogo de estudos a respeito de bancos de dados não-relacionais é extenso, com diversos artigos analisando sua eficiência, escalabilidade e adequação a diferentes cenários de uso. É possível encontrar tanto materiais que focam em analisar bancos de dados específicos quanto fontes que buscam comparações entre os diferentes tipos disponíveis no mercado.

Para uma visão geral sobre bancos de dados NoSQL, seus diferentes tipos e usabilidade, Bathla, Rani e Aggarwal (2018) tratam sobre características de cada uma dessas tecnologias, e qual delas é mais apropriada para cumprir diferentes requisitos. São analisados diversos desses bancos, como BigTable, Cassandra, HBase, Neo4J, OrientDB, DynamoDB, Oracle NoSQL, MongoDB e CouchDB.

Xiao e Liu (2011) tratam da utilização do banco de dados NoSQL HBase para armazenamento de imagens de um sistema de informações geográfica. É apresentada uma visão geral de sua arquitetura e destacadas suas características que permitiram a resolução de problemas de gerenciamento de dados enfrentados previamente pelos autores, como os diferentes formatos de dados disponíveis e falta de desempenho ao lidar com grandes volumes de dados.

Dois anos depois, Liu et al. (2013) expandem a análise, analisando a performance do HBase para diferentes cargas de trabalho sob os aspectos de importação e processamento de imagens. São testados cenários de tamanhos variados e com quantidades diferentes de nodos, com a conclusão de que o aumento do cluster do HBase incorre em aumento do poder de processamento, solidificando a escalabilidade e viabilidade da utilização desse banco de dados NoSQL para armazenamento e processamento de imagens.

Ainda no ramo do sensoriamento remoto, Hajjaji e Farah (2018) investigam o desempenho de três bancos de dados NoSQL: Cassandra, HBase e MongoDB. Utilizando grupos com diferentes quantias e tamanhos de imagens, o objetivo era testar a performance desses sistemas ao escalar o tamanho dos dados e o tamanho dos clusters. A partir da obtenção de dados a respeito do aumento de velocidade e da eficiência de cada um, os autores não só verificaram a viabilidade do uso desses bancos de dados para o armazenamento das imagens de sensoriamento remoto, como concluíram que o Cassandra apresentou os melhores resultados.

Hein e Blankenbach (2021) apresentam o uso do Neo4J para o armazenamento de metadados de imagens *raster*, imagens definidas como uma matriz de valores, de análise geoespacial. Os autores citam que a decisão de armazenar apenas os metadados diretamente no Neo4J parte do fato de que outros trabalhos indicam que guardar dados binários diretamente no banco de dados não é eficiente, e é isso que desejamos explorar nesse trabalho.

Outra área de pesquisa com grandes contribuições para o tema foi a de imagens médicas. Nagappa e Divya (2017) apresentam uma análise concisa do histórico do uso de bancos de dados SQL para armazenamento dessas, em seguida destacando vantagens do uso de NoSQL e fornecendo explicações a respeito dessa tecnologia, usando como exemplo o MongoDB e o CouchDB. Um dos trabalhos citados nesse estudo é o de Bastião Silva et al. (2014), onde os autores destrincham o padrão de armazenamento de imagens médicas, e comparam a eficiência do MongoDB, CouchDB e Lucene (esse último sendo capaz apenas de guardar os metadados, e não arquivos binários), com e sem a utilização do método de *file system*, que armazena as imagens de forma similar à utilizada no próprio sistema de arquivos. Ao final, foi possível observar que o desempenho dos dois primeiros foi similar, com o MongoDB apresentando rapidez de armazenamento e leitura, e o CouchDB sendo destacado para casos em que todos os tipos de consulta são previamente conhecidos.

Rebecca e Shanthi (2016a) apresentam uma interessante comparação entre MongoDB e MySQL, comparando o tempo de resposta de ambos para armazenamento e leitura de imagens de diferentes tamanhos e com processadores i3 e i5 (de geração não especificada). Para quase todos os casos de teste, o MongoDB apresentou tempos de resposta significativamente menores.

Seis meses depois, Rebecca e Shanthi (2016b) apresentam nova pesquisa, dessa vez comparando o tempo de resposta de armazenamento e leitura de dois bancos de dados NoSQL: MongoDB e Cassandra. Os resultados encontrados foram interessantes, uma vez que o Cassandra demonstrou respostas mais demoradas de maneira diretamente proporcional ao tamanho dos dados, enquanto o MongoDB obteve maior consistência. A conclusão das autoras foi de que, embora ambos sejam adequados para o armazenamento de imagens médicas, uma vez que o MongoDB apresenta melhores resultados para dados de maior tamanho, este seria o melhor candidato.

O comportamento estável do MongoDB para grandes volumes pode ser observado também no estudo feito por Boicea, Radulescu e Agapin (2012), onde os autores buscaram uma comparação entre este e o banco de dados SQL Oracle, onde durante os casos de teste de inserção, deleção e atualização, é possível observar a boa estabilidade do MongoDB, enquanto o Oracle apresenta tempos de resposta cada vez maiores.

Ainda nas comparações diretas entre dois bancos de dados, Chopade e Dhavase (2017) testam MongoDB e Couchbase com operações de leitura e inserção de imagens. Os resultados atestam a otimização do MongoDB para trabalhar com leitura, conforme explicado no Manual do MongoDB (2023), além de mostrar o melhor desempenho do Couchbase para operações de escrita.

Comparações a respeito da eficiência de diversos bancos de dados NoSQL são de extrema importância, mesmo que os testes não sejam feitos com imagens, pois nos fornecem um panorama geral a respeito da performance destes. Isso foi realizado por Martins, Ab-

basi e Sá (2019), que utilizando cargas com diferentes distribuições de operações de leitura, escrita e atualização, compararam os bancos de dados MongoDB, Redis, Memcached, OrientDB, Voldemort, Cassandra e HBase. Resultados interessantes foram observados, como por exemplo, o MongoDB apresentando tempos de execução altos devido aos mecanismos de trava acionados durante operações de atualização. A conclusão nos apresenta diversos pontos importantes, como Redis, Memcache e Voldemort com o melhor desempenho, mas sacrificando consistência de dados para tal, Cassandra e HBase se destacando para operações de atualização e MongoDB mantendo sua boa performance para quantidades elevadas de dados, conforme visto em outros estudos.

Mukherjee (2019) também realiza uma extensiva análise teórica sobre a estrutura de bancos de dados NoSQL e vários dos seus exemplos.

Oliveira Assis et al. (2017) fizeram um interessante trabalho que agrega comparações de bancos de dados NoSQL entre si e com um SQL também. Na primeira parte do trabalho, avaliando os bancos Redis, MongoDB e Cassandra com duas cargas de trabalho, uma igualmente dividida entre leitura e escrita e outra mais focada em leituras, os autores puderam observar o alto fluxo de operações por segundo e a baixa latência do Redis, além de observar a otimização na escrita do Cassandra, como explicado por Cooper et al. (2010).

Já na segunda parte, MongoDB e MySQL são comparados no armazenamento de mídias, alvo do nosso trabalho, e tal qual na pesquisa realizada por Győrödi et al. (2015), é possível atestar, a partir dos resultados de tempo de resposta de leitura e escrita muito melhores, a significativa vantagem de se utilizar o MongoDB ao invés da solução SQL contra o qual foi testado.

É tentador imaginar que os NoSQL são uma tecnologia simplesmente superior, porém esse não é o caso. Embora não tratem de arquivos de mídia, Li e Manoharan (2013) fazem uma comparação entre seis bancos de dados não-relacionais, sendo eles MongoDB, RavenDB, CouchDB, Cassandra, Hypertable e Couchbase, com um banco de dados relacional, o Microsoft SQL Express. Os sete foram testados para operações de instanciar o *bucket* (o armazenamento flexível de recursos da Amazon), leitura, escrita, deleção e uma operação especial denominada pelos autores de "buscar todas as chaves". Os resultados foram bem variados, e vale citar os casos notáveis como a operação de criar instância, onde o Microsoft SQL apresentou o pior resultado dentre todos, e o de buscar todas as chaves, onde o banco de dados relacional teve o melhor desempenho. Nos outros três testes, as posições foram variadas.

4 Metodologia Proposta

Este capítulo discorrerá a respeito da metodologia proposta para realização deste trabalho, detalhando a escolha dos bancos de dados, os testes a serem realizados, o ambiente em que ocorrerão os testes, como será feita a captura de resultados para posterior avaliação no Capítulo 5 e o que esperamos obter em cada um desses passos.

4.1 Escolha dos Bancos de Dados

4.1.1 Bancos de Dados SQL

Para a escolha dos bancos de dados relacionais, foram avaliados os seguintes critérios, definindo uma restrição de domínio tanto da aplicação, quanto da implementação utilizada, a saber:

- Possibilidade de uso do banco de dados via imagem Docker;
- Suporte para bibliotecas em Python;
- Popularidade no mercado;
- Uso do banco de dados em artigos científicos do estado da arte;

Para o primeiro critério, utilizamos como fonte primária o site Docker Hub (2023), que contém milhares de imagens Docker de vários bancos de dados, plataformas, *frameworks*, etc.

Para o suporte de bibliotecas conectoras em Python, foram utilizadas as documentações oficiais e bibliotecas conhecidas para cada um dos bancos de dados analisados. Ambos os critérios buscam estabelecer uma padronização no método de acesso e de uso dos bancos de dados.

Para a busca de uso de mercado, uma vez que não foi identificada uma entidade central que fornecesse esse tipo de informação, foi utilizado o *ranking* disponibilizado pela plataforma de dados de mercado e estatísticas Statista (2022), que inclui na pesquisa tanto bancos de dados relacionais quanto não-relacionais, bem como as pesquisas anuais realizadas no Stack Overflow (2022). Já para encontrar os usos dos bancos de dados em artigos científicos, utilizamos as ferramentas IEEE Xplore (2023), ResearchGate (2023) e ScienceDirect (2023).

Após análise dos critérios, foram escolhidos os seguintes bancos de dados relacionais para nossa pesquisa:

- MariaDB (2023a)
- Microsoft SQL Server (2023)
- MySQL (2023)
- PostgreSQL (2023b)

Fora os bancos de dados relacionais escolhidos, alguns outros foram considerados para a pesquisa. Dentre eles, vale a pena citar o DB2, da IBM (2023), o Oracle, da Oracle (2023), e o Aurora, da Amazon AWS (2023a).

4.1.2 Bancos de Dados NoSQL

Seguindo o definido no item anterior, para a escolha dos bancos de dados não-relacionais, foram avaliados os seguintes critérios:

- Diversidade de modelos;
- Possibilidade de uso do banco de dados via imagem Docker;
- Suporte para bibliotecas em Python;
- Popularidade no mercado;
- Quantidade de artigos científicos que utilizam o banco de dados;

Os critérios utilizados foram os mesmos para os bancos de dados relacionais, com adição do critério de diversidade de modelos. Como explicado no Capítulo 2, bancos de dados não-relacionais possuem diversos tipos de modelagem de dados, como chave-valor, documento, etc. Para essa pesquisa, buscamos uma boa variedade de modelos a fim de conseguir extrair informações de diferentes arquiteturas.

Ao final, foram escolhidos os seguintes bancos de dados não-relacionais:

- Cassandra (2023), modelo de colunas;
- MongoDB (2023b), modelo de documentos;
- Redis (2023b), modelo chave-valor;
- Solr (2023a), modelo de documentos;

Novamente, fora os escolhidos, outros bancos de dados não relacionais foram considerados para a pesquisa. Dentre eles, citamos o ArcadeDB (2023) e o S3, da Amazon AWS (2023b).

4.2 Detalhamento dos Bancos de Dados

4.2.1 MariaDB

MariaDB é um banco de dados SQL de código aberto que surgiu originalmente como um *fork* do MySQL, ou seja, foi baseado no código fonte do MySQL. Atualmente, sua licença é a *GPL License v2*[1].

Em uma página do MariaDB (2023b), podemos ver como algumas de suas características ser compatível com MySQL, além de possuir uma configuração leve, uma grande comunidade e a possibilidade de integração com outros mecanismos de armazenamento. Além disso, oferece suporte ao armazenamento de dados orientado à linhas e a colunas.

O MariaDB conta com suporte para integração com C, C++, Java, Node.js, ODBC e Python. Para a integração com Python que buscamos, utilizaremos a biblioteca *mariadb*.

4.2.2 Microsoft SQL Server

Microsoft SQL Server é um banco de dados SQL de código proprietário, de propriedade da Microsoft e sob sua licença[2]. De acordo com a página da Microsoft (2023b), o SQL Server 2019 apresenta alta escalabilidade e desempenho, além de integração com outras ferramentas como o Power BI para análise de dados.

Na documentação provida pela Microsoft (2023a), podemos encontrar também informações a respeito do uso do SQL Server para tratamento de *Big Data*. Com suporte para integração com C#/NET, ADO, Java, Node.js, ODBC, PHP, Python, Ruby e Spark, utilizaremos a biblioteca *pyodbc* para nossa integração com Python.

4.2.3 MySQL

MySQL é um banco de dados SQL com versões de código aberto (*Community Edition*) e de código proprietário (*Enterprise Edition*) da Oracle, sendo a licença da *Community Edition* a *GPL License*[3]. De acordo com os dados da *2022 Developer Survey* do Stack Overflow (2022), foi naquele ano o banco de dados mais usado entre usuários que estão aprendendo a programar e entre usuários em geral.

De acordo com a página da Oracle Cloud (2023), o MySQL é de fácil acesso e atingiu grande maturidade após ser testado em diversos sistemas ao longo dos anos. Com suporte para integração com C, C#/NET, C++, Delphi, Java, Node.js, ODBC, PHP, Perl, Python e Ruby, utilizaremos a biblioteca *pymysql* para integração com Python.

[1] https://www.gnu.org/licenses/old-licenses/gpl-2.0.html
[2] https://www.microsoft.com/en-us/licensing/product-licensing/sql-server
[3] https://www.gnu.org/licenses/gpl-3.0.en.html

4.2.4 PostgreSQL

PostgreSQL é um banco de dados SQL de código aberto, sendo sua licença a *PostgreSQL License*[4]. De acordo com os dados da *2022 Developer Survey* do Stack Overflow (2022), foi considerado o banco de dados mais amado e desejado pelos desenvolvedores naquele ano, ultrapassando o então campeão Redis.

De acordo com a documentação do PostgreSQL (2023a), essa reputação se dá pela segurança, integridade e escalabilidade provida. Tendo diversas extensões providas pela comunidade. O PostgreSQL também permite ao usuário definir tipos próprios para seus dados, criar funções e integrar com diversas linguagens. Com suporte para integração com C#/.NET, C++, Delphi, Java, Node.js, Perl, PHP e Python, utilizaremos a biblioteca *psycopg2* para integração com Python.

4.2.5 Cassandra

Cassandra é um banco de dados NoSQL de código aberto com arquitetura distribuída que implementa o modelo de colunas, sendo sua licença a *Apache License 2.0*[5]. Utilizando um sistema de nodos e *clusters*, ele é capaz de atingir alto nível de escalabilidade e replicabilidade, e possui também a CQL (do inglês, *Cassandra Query Language* - Linguagem de Consulta do Cassandra), similar ao SQL, mas adaptada para as especificidades do Cassandra, conforme mostra o artigo da SpiceWorks (2023).

O modelo colunar, conforme explicitado pelo artigo da AWS (2023a), é criado para aumentar a escalabilidade horizontal, o que o torna ideal para processamento de *Big Data*, ou seja, ideal para dados não estruturados, como imagens. Com suporte para integração com C#/NET, C++, Clojure, Dart, Elixir, Erlang, Go, Haskell, Java, Node.js, PHP, Perl, Python, Ruby, Rust e Scala, utilizaremos a biblioteca *cassandra-driver* para integração com Python.

Outra funcionalidade interessante do Cassandra é a existência de *prepared statements*, que de acordo com a documentação do driver disponibilizada pela DataStax (2023b), são *queries* que podem ser salvas para uso posterior quando a única alteração a ser feita são seus parâmetros. O uso dessa funcionalidade diminui o tráfego de rede e a utilização de CPU, o que é de grande utilidade para os testes desse trabalho, que utilizam *queries* repetitivas com alteração apenas de parâmetros.

4.2.6 MongoDB

MongoDB é um banco de dados NoSQL de código aberto de alto desempenho que implementa o modelo de documentos. Sua licença é a *Server Side Public License (SSPL)*[6], que

[4] https://www.postgresql.org/about/licence/
[5] https://www.apache.org/licenses/LICENSE-2.0
[6] https://www.mongodb.com/licensing/server-side-public-license

é alvo de constantes debates, como pode ser observado em um artigo de Pete Scott (2023). É o banco de dados que mais atrai usuários iniciantes do MySQL e o NoSQL mais desejado, conforme mostram os dados da *2022 Developer Survey* do Stack Overflow (2022).

Com foco na escalabilidade horizontal, o armazenamento dos dados é feito a partir de um *schema* flexivel para documentos chamado BSON, uma representação binária de dados que pode ser interpretada como um JSON por outras aplicações, como explicado em um artigo oficial publicado na página do MongoDB (2023c). Essa estrutura facilita o armazenamento tanto de dados estruturados como não-estruturados, esse sendo o assunto principal desse trabalho.

Além disso, com o avanço da tecnologia, o tamanho e qualidade das fotos tende a aumentar, e se mostra necessário que os bancos de dados consigam acompanhar esse crescimento. Nesse contexto, é importante citar o GridFS, uma especificação para armazenamento e leitura de imagens maiores do que o limite estabelecido para BSONs de 16MB, explicitado na documentação do MongoDB (2023a). O GridFS não será explorado nesse trabalho, mas é de extrema relevância para o assunto.

O MongoDB tem suporte para integração com diversas linguagens, como C, C++, C#/NET, Go, Java, Node.js, PHP, Python, Ruby, Rust, Scala e Swift. Para nossa pesquisa, utilizaremos a biblioteca *pymongo* para realizar a integração com Python.

4.2.7 Redis

Redis é um banco de dados NoSQL de código aberto que implementa o modelo chave-valor, sendo sua licença a *BSD-3-Clause*[7]. De acordo com as pesquisas anuais do Stack Overflow (2022), foi considerado o banco de dados mais amado pelos desenvolvedores por cinco anos seguidos, até ser superado pelo PostgreSQL em 2022.

De acordo com a documentação do Redis (2023a), ele é um armazenador de dados em memória. A persistência desses pode ser feita tanto através de escritas periódicas ao disco quanto por meio de um log de ações, que ao serem executadas novamente, permitem a reconstrução do banco de dados quando esse é iniciado novamente.

Tal característica contrasta com bancos de dados que armazenam as informações diretamente em disco rígido ou SSDs. Esse tipo de arquitetura confere maior performance, pois consegue minimizar os tempos de resposta ao remover o tempo necessário para acessar o disco, porém ao custo de maior risco, uma vez que um erro poderia levar à perda de dados, conforme explicado na página da AWS (2023b).

Além disso, o Redis possui integração com diversas linguagens, como C#/NET, Go, Java, Node.js e Python. Para esse trabalho, foi utilizada a biblioteca *redis-py* para integração com Python, conforme orientado na documentação.

[7] https://opensource.org/license/bsd-3-clause/

4.2.8 Solr

Solr é um banco de dados NoSQL de código aberto que implementa o modelo de documentos, sendo sua licença a *Apache License 2.0*[8]. De acordo com a página do Solr (2023b), ele é otimizado para larga escala e extremamente veloz, suportando configurações com e sem estruturas definidas.

Com alta performance e suporte para arquivos JSON, XML, CSV e binários via HTTP, além de suporte a diversos plugins, é um banco de dados robusto, com grande potencial para o armazenamento de imagens. Sua integração é feita a partir de uma RESTful API, e suporta diversas linguagens. Para a integração com Python, utilizaremos a biblioteca *pysolr*.

4.3 Configurações Utilizadas

Para mantermos ambas simplicidade e padronização, fizemos a instalação padrão das imagens Docker, sempre dando preferência à imagens oficiais do Dockerhub. Durante a instalação, foram adicionados apenas o Python, necessário para rodar os *scripts*, e o editor de textos nano, para facilitar a edição dos *scripts*.

Embora os bancos de dados estudados apresentem suporte para trabalho com *clusters* e customizações que auxiliam na otimização, isso introduziria diferenças muito grandes entre os bancos, que poderiam causar interferência na comparação dos resultados. Dessa forma, optamos por não utilizar essas configurações.

4.4 Definição dos *Benchmarks*

Para que seja possível avaliar os resultados, precisamos antes definir quais serão os parâmetros avaliados durante os testes para determinar o desempenho de cada um dos bancos de dados:

- **Tempo de resposta das operações de inserção, deleção e leitura**: como bancos de dados costumam ser usados por milhares de usuários ao mesmo tempo, ter um baixo tempo de resposta é crucial para a aplicação, conforme explicado por Elmasri e Navathe (2016, p. 304–328). Dessa forma, é o principal parâmetro a ser avaliado, obtenção de valores explicada na seção de Método de Coleta de Dados;
- **Desvio padrão das medidas de tempo de resposta**: analisando o desvio padrão em cada caso de teste, podemos ter uma ideia da distribuição dos tempos de resposta, nos dando indícios de quais bancos de dados apresentam maior estabilidade;

[8] https://www.apache.org/licenses/LICENSE-2.0

- **Variação da resposta em função do *hardware***: com esse parâmetro será possível saber quais dentre os bancos de dados estudados fazem melhor uso dos recursos de hardware disponíveis, além de revelar também quais BDs exigem maior consumo da máquina;

4.5 Método de Coleta de Dados

Para coletar os dados, é necessária uma ferramenta que seja capaz de trabalhar tanto com bancos de dados relacionais quanto não-relacionais, além de precisar ser aplicável à quaisquer tecnologias do gênero, dado o uso de alguns bancos de dados para esse estudo que não são muito comuns.

Foram considerados o uso do SolarWinds (2023) ou do *Yahoo! Cloud Serving Benchmark*, de Cooper (2023), mas as métricas escolhidas, bem como a possibilidade de inserção de sobrecargas por conta das ferramentas inviabilizaram sua utilização.

Dessa forma, optamos por criar um *script* em Python, adaptado para cada um dos bancos de dados utilizados, que orquestrasse:

- A criação do banco de dados e das tabelas necessárias;
- A execução das operações nos bancos de dados, conforme especificações de teste;
- Coleta das métricas e exportação destas para um arquivo CSV para posterior análise;
- Deleção do banco de dados e recomeço do processo.

Para a coleta dos tempos necessários em cada operação, foi utilizada a biblioteca *timeit*, nativa do Python, que ignora desvios de tempo por processos de sistema. Na orquestração, a execução do comando é ainda separada com o uso de um mutex, que isola a utilização do recurso. O código redigido foi devidamente alterado para ser compatível com cada um dos bancos de dados, e todos podem ser encontrados na página do GitHub do projeto, de Miranda e Albuquerque (2023).

Com o objetivo de metrificar apenas as otimizações internas de cada banco de dados na execução, todas as conexões são feitas em modo local, retirando possíveis interferências de uma conexão remota. Também não é analisado o impacto de cada operação quando feita de modo remoto.

4.6 *Workloads* de Teste

A base de imagens utilizada para realizar os testes desse trabalho é uma parte da ILSVRC2015 (2023), com a adição de fotos de autoria própria para dar maior variedade nos

tamanhos de imagens do conjunto de dados de testes. Foram escolhidas 3010 imagens da base ILSVRC2015 e 290 imagens próprias, de maneira a não ultrapassar os recursos disponíveis das máquinas utilizadas para o teste, que serão explicitadas na seção de Especificações das Máquinas dessa pesquisa. A relação das fotos pode ser encontrada na página do GitHub do projeto, de Miranda e Albuquerque (2023).

Os *workloads* definidos para esse estudo seguem a estrutura apresentada no fluxograma na Figura 4.4. As operações utilizadas serão inserção (única e em rajada), leitura e deleção das imagens. Para evitar que o cache do servidor onde os testes são rodados tenha grande interferência nos resultados, todos os testes são realizados trinta vezes para cada operação, sendo os resultados do trigésimo teste utilizados para a análise, de maneira a minimizar o impacto da variância de tempo por *caching* das imagens na coleta de medidas.

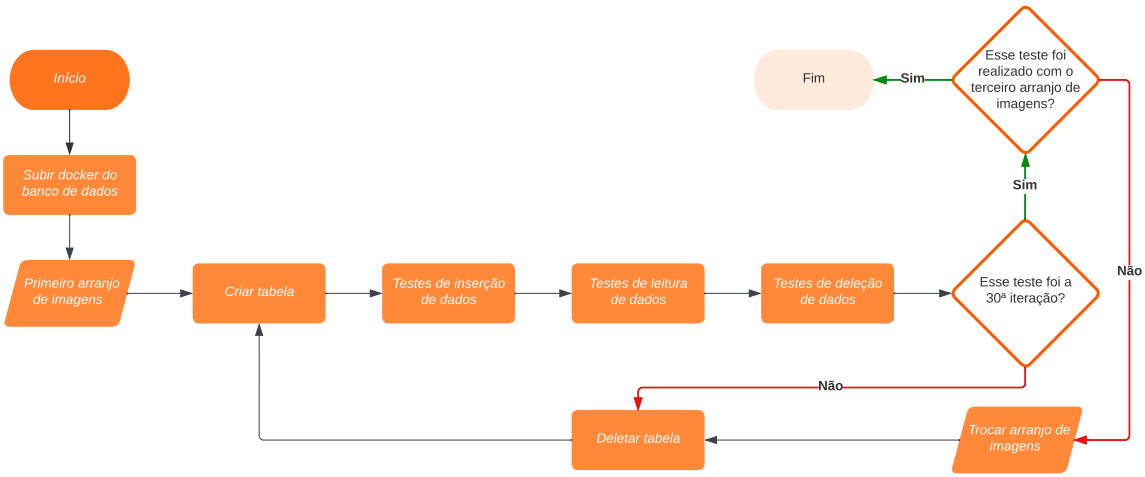

Figura 4.4 – Fluxograma de execução dos *workloads*

Fonte: Produzido pelos autores.

Cada um dos conjuntos de testes será realizado com três diferentes ordenações para as mesmas imagens:

- Ordem crescente, de maneira a medir o impacto do aumento gradual do tamanho dos arquivos a serem armazenados;
- Ordem decrescente, de maneira a medir o impacto da diminuição gradual do tamanho dos arquivos a serem armazenados;
- Ordem aleatória, de maneira a inspecionar o comportamento dos bancos de dados ao lidar com mudanças bruscas e sem padrão pré-definido do tamanho das imagens.

Durante os testes, as máquinas utilizadas não operarão de maneira sobrecarregada. Isso será feito a fim de evitar que o gerenciamento de recursos disponíveis, que é feito automaticamente pela provedora das máquinas, não interfira nos resultados.

Com tudo isso apresentado, e a fim de testar o impacto de *queries* únicas e em rajada, foram determinados os seguintes cenários de teste:

- **Operação de inserção única para 3300 imagens**;

Nesse teste poderemos avaliar o tempo de resposta do sistema ao inserir uma única imagem no banco de dados, nos dando uma noção inicial de quais bancos de dados possuem maior vantagem ou desvantagem em relação aos outros na hora de armazenar dados binários.

- **Operação de inserção em rajadas de 3 imagens para 3300 imagens**;

Com a inserção em rajadas, desejamos testar o comportamento do banco de dados ao processar uma maior volume de dados em uma menor quantidade de operações. Ao comparar com os resultados de inserção única, poderemos observar se há maior eficácia em muitos processamentos de menor tamanho, ou em poucos processamentos de maior tamanho.

- **Operação de leitura para 3300 imagens**;

Sendo a leitura do banco de dados a operação que devolve ao usuário o resultado esperado, essa é uma operação de grande interesse para os testes, pois nos permite avaliar o tempo de resposta a um usuário de um sistema de imagens armazenadas em banco de dados.

- **Operação de deleção para 3300 imagens**;

Ao avaliar o tempo de resposta da deleção, podemos observar como cada banco de dados se comporta ao tentar apagar dados de tamanho elevado para abrir espaço para outras inserções.

5 Resultados

Com as medidas obtidas para todos os bancos de dados em estudo, é possível criar comparações entre os sistemas e métodos de armazenamento de dados.

5.1 Especificações das Máquinas

Para estabelecermos uma base consistente para que os testes pudessem ser comparados, se mostrou necessário minimizar a interferência do sistema na execução dos casos de teste.

Inicialmente, os testes seriam realizados em um servidor disponibilizado pela UnB. Foi notada, porém, grande interferência no ambiente, uma vez que diversas rotinas de sistema eram constantemente executadas. Uma vez que essas flutuações comprometeriam os resultados finais, foi necessária outra solução.

Com isso em mente, e sendo de interesse testar a influência do *hardware* na execução das operações, foram utilizadas duas máquinas AWS com dedicação exclusiva, cujas especificações estão dispostas na Tabela 5.2:

Tabela 5.2 – Especificações de *hardware* das máquinas AWS utilizadas

Instância	vCPU	Mem (GiB)	Mem GPU (GiB)	*Bandwidth* (Gbps)	Processador
p3.2xlarge	8	61	16	1,5	Intel Xeon E5-2686 v4 2.3GHz
p3.8xlarge	32	244	64	7	Intel Xeon E5-2686 v4 2.3GHz

Fonte: Amazon AWS (2023)

A fim de aferir tais dados na prática, utilizamos o comando **htop** para verificar as especificações da máquina, e os resultados podem ser observados na Figura 5.5 e na Figura 5.6.

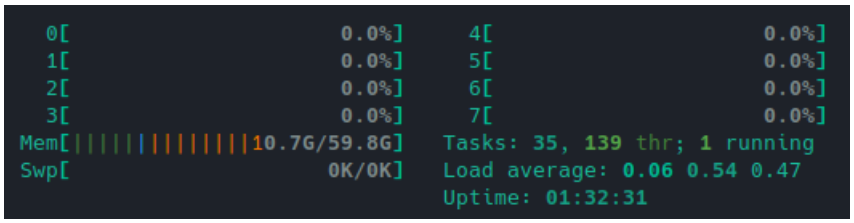

Figura 5.5 – Especificações da máquina p3.2xlarge obtidas através do terminal

Fonte: Produzido pelos autores.

Ambas as máquinas possuem o Ubuntu 22.04 como sistema operacional, e uma capacidade de armazenamento de 100GiB com o uso de SSDs.

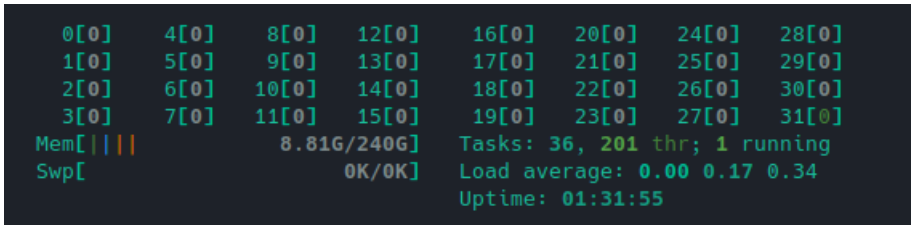

Figura 5.6 – Especificações da máquina p3.8xlarge obtidas através do terminal

Fonte: Produzido pelos autores.

5.2 Bancos de dados SQL

Após a conclusão dos testes, analisamos o tempo médio da operação em segundos (t_m) e o desvio padrão (σ) para cada um dos bancos da dados testados, levando em conta a ordenação das imagens e em qual máquina ocorreu a execução dos testes.

Além disso, analisamos também os valores de tempo de execução mínimo e máximo de cada conjunto de operações.

5.2.1 Testes de inserção única

A Tabela 5.3 mostra os resultados, medidos em segundos, obtidos para todos os casos testados, destacando, para cada ordenação, os melhores resultados em azul e os piores em vermelho. A Tabela 5.4, por sua vez, dispõe a mudança percentual das medidas feitas na máquina p3.8xlarge em relação à p3.2xlarge.

Tabela 5.3 – Resultados dos testes de inserção singular dos bancos de dados relacionais

Banco de dados e resultados		Ordenação de imagens e máquina utilizada					
		Crescente		Decrescente		Aleatório	
		2xlarge	8xlarge	2xlarge	8xlarge	2xlarge	8xlarge
MariaDB	t_m (s)	0.006431	0.006406	0.006661	0.006696	0.006260	0.006197
	σ (s)	0.028344	0.029877	0.030028	0.027288	0.022759	0.023788
Microsoft SQL	t_m (s)	0.009206	0.008737	0.008793	0.009039	0.009255	0.008802
	σ (s)	0.035312	0.040911	0.029130	0.019009	0.019309	0.023120
MySQL	t_m (s)	0.052881	0.049131	0.050822	0.051711	0.051290	0.053107
	σ (s)	0.118981	0.114785	0.119361	0.120330	0.119118	0.120006
PostgreSQL	t_m (s)	0.019183	0.018283	0.020249	0.018147	0.019063	0.018561
	σ (s)	0.055568	0.049056	0.057613	0.049080	0.050761	0.048243

Fonte: Produzido pelos autores.

Tabela 5.4 – Mudança percentual dos resultados ao trocar da 2xlarge para a 8xlarge (inserção singular, SQL)

Banco de dados e mudança percentual		Ordenação de imagens		
		Crescente	Decrescente	Aleatório
MariaDB	t_m%	-0.39%	+0.53%	-1.01%
	σ%	+5.41%	-9.12%	+4.52%
Microsoft SQL	t_m%	-5.09%	+2.80%	-4.89%
	σ%	+15.86%	-34.74%	+19.74%
MySQL	t_m%	-7.09%	+1.75%	+3.54%
	σ%	-3.53%	+0.81%	+0.75%
PostgreSQL	t_m%	-4.69%	-10.38%	-2.63%
	σ%	-11.72%	-14.81%	-4.96%

Fonte: Produzido pelos autores.

A Tabela 5.3 já mostra a facilidade do MariaDB em realizar essas operações. Explicitando as diferenças na Tabela 5.4, temos o PostgreSQL com a maior redução no tempo médio na troca de máquinas (redução de 10.38%) e o MicrosoftSQL com a maior redução no desvio padrão (redução de 34.74%).

Para observar a distribuição dos dados, a Figura 5.7 apresenta as medições de tempo de resposta em ordenação crescente na máquina 2xlarge dispostos em um gráfico do tipo violino. Para as outras ordenações e máquinas, as Figuras A.46, A.47, A.48, A.49 e A.50 podem ser consultadas no Apêndice A.

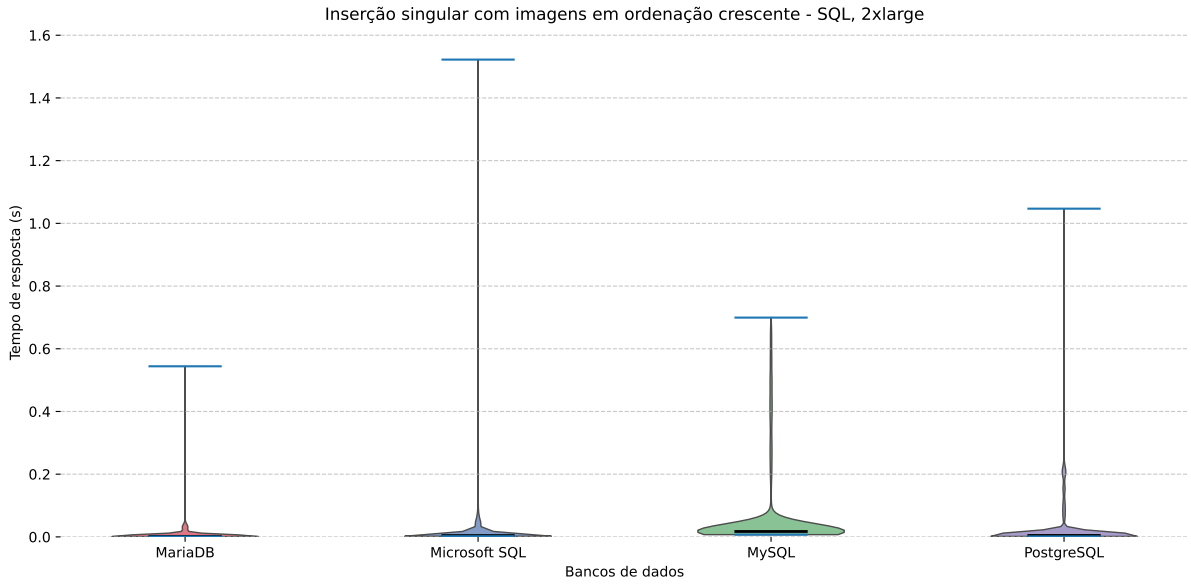

Figura 5.7 – Gráfico dos tempos de resposta (inserção singular, crescente, SQL, 2xlarge)

Fonte: Produzido pelos autores.

O MariaDB apresentou, em geral, os melhores resultados dentre os bancos da dados relacionais, ao passo que o MySQL apresentou os piores. Um fator notável foi que, embora o MariaDB tenha sido o banco de dados com menor influência da troca de hardware no tempo

médio de operação, o que sofreu menor impacto no grau de variação de seus dados foi o MySQL. Por outro lado, o Microsoft SQL teve variação considerável de seu desvio padrão nas trocas de máquina, indicando uma falta de estabilidade em relação à variação de hardware.

Dos quatro testados, o PostgreSQL foi o único que consistentemente diminuiu seu tempo médio de execução ao dispor de uma máquina mais robusta. Embora a variação dos outros possa ter sido influenciada por flutuações de difícil controle no ambiente, é um resultado indicativo de uma proporcionalidade direta entre a eficiência do PostgreSQL e a qualidade do hardware.

Tabela 5.5 – Casos com pior e melhor resultado para cada banco de dados (inserção singular, SQL)

BD	Pior tempo médio	Melhor tempo médio	Menor distribuição	Maior distribuição
MariaDB	Decrescente - 8x	Aleatório - 8x	Decrescente - 2x	Aleatório - 2x
Microsoft SQL	Aleatório - 2x	Crescente - 8x	Decrescente - 8x	Crescente - 8x
MySQL	Aleatório - 8x	Crescente - 8x	Crescente - 8x	Decrescente - 8x
PostgreSQL	Decrescente - 2x	Decrescente - 8x	Aleatório - 8x	Decrescente - 2x

Fonte: Produzido pelos autores.

Avaliando a Tabela 5.5, é interessante notar que a ordenação crescente não só não apresentou o pior tempo médio para nenhum dos bancos de dados, como foi o caso com o melhor tempo em dois deles.

De posse do menor e maior valor de tempo de resposta para cada caso, foram elaborados os gráficos das Figuras 5.8 e 5.9.

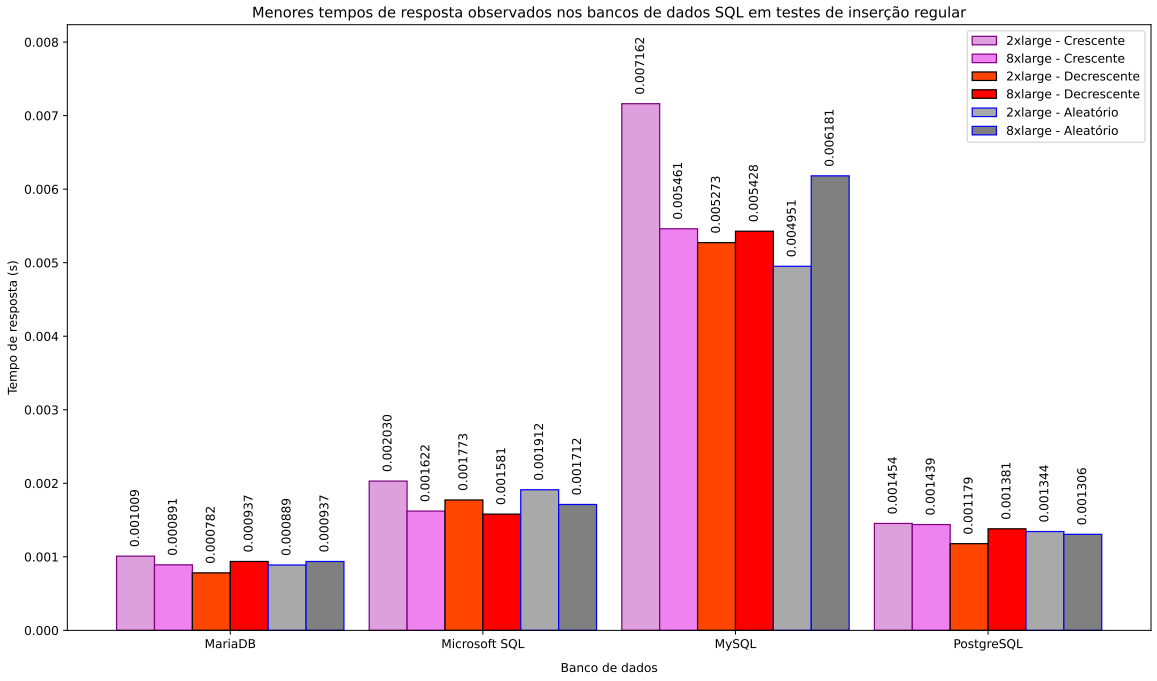

Figura 5.8 – Gráfico do menor tempo de resposta de inserção única em SQL

Fonte: Produzido pelos autores.

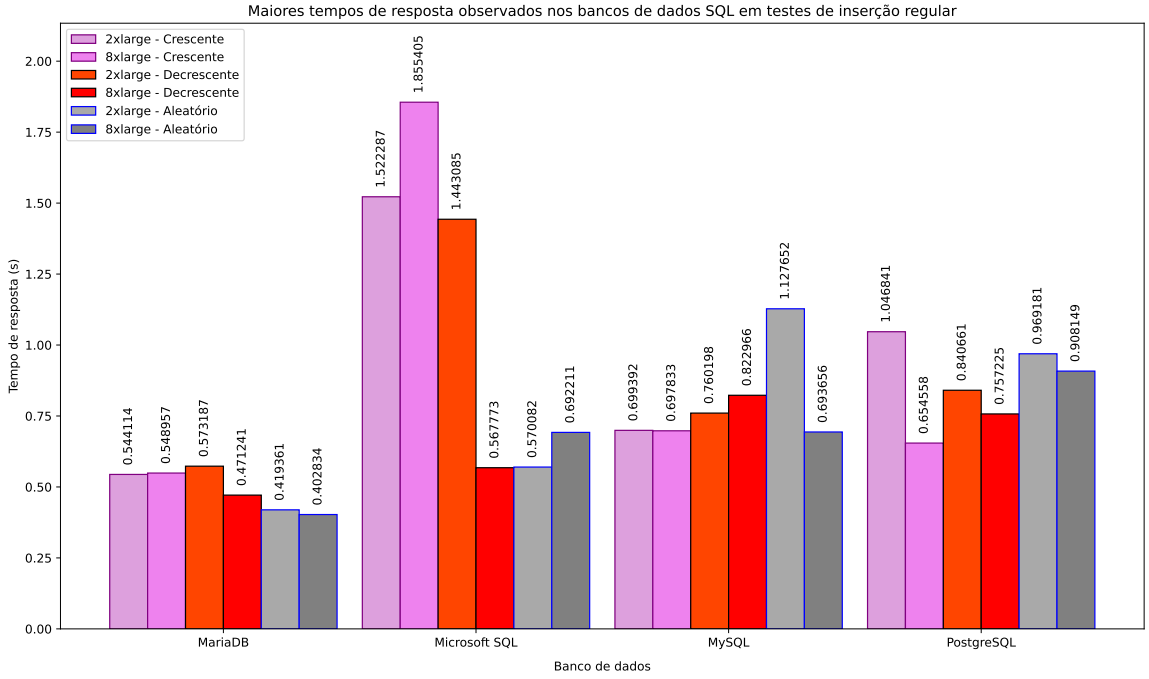

Figura 5.9 – Gráfico do maior tempo de resposta de inserção única em SQL

Fonte: Produzido pelos autores.

É possível notar, novamente, que o MariaDB apresentou os melhores resultados em ambos os casos.

O Microsoft SQL apresentou os maiores picos de tempo de resposta, enquanto o MySQL teve os resultados mais elevados para os valores mínimos, indicando seu baixo desempenho para armazenamento de imagens quando comparado aos outros bancos de dados relacionais.

5.2.2 Testes de inserção em rajadas de 3 imagens

A Tabela 5.6 mostra os resultados obtidos para todos os casos testados, destacando os melhores resultados de cara ordenação em azul e os piores em vermelho, e a Tabela 5.7 dispõe a mudança percentual das medidas feitas em cada máquina.

Tabela 5.6 – Resultados dos testes de inserção em rajadas dos bancos de dados relacionais

Banco de dados e resultados		Ordenação de imagens e máquina utilizada					
		Crescente		Decrescente		Aleatório	
		2xlarge	8xlarge	2xlarge	8xlarge	2xlarge	8xlarge
MariaDB	t_m (s)	0.017987	0.018219	0.018087	0.018188	0.018375	0.018746
	σ (s)	0.068914	0.072373	0.069378	0.067443	0.058862	0.057521
Microsoft SQL	t_m (s)	0.033539	0.032120	0.032164	0.030669	0.033131	0.032420
	σ (s)	0.086603	0.083798	0.082516	0.080436	0.045360	0.049187
MySQL	t_m (s)	0.136303	0.136382	0.135565	0.131621	0.142807	0.135192
	σ (s)	0.343933	0.339943	0.343023	0.333469	0.199537	0.191531
PostgreSQL	t_m (s)	0.051306	0.050993	0.051113	0.050797	0.051772	0.051722
	σ (s)	0.146205	0.146778	0.140138	0.136485	0.076981	0.076195

Fonte: Produzido pelos autores.

Tabela 5.7 – Mudança percentual dos resultados ao trocar da 2xlarge para a 8xlarge (inserção em rajadas, SQL)

Banco de dados e mudança percentual		Ordenação de imagens		
		Crescente	Decrescente	Aleatório
MariaDB	t_m%	+1.29%	+0.56%	+2.02%
	σ%	+5.02%	-2.79%	-2.28%
Microsoft SQL	t_m%	-4.23%	-4.65%	-2.15%
	σ%	-3.24%	-2.52%	+8.44%
MySQL	t_m%	+0.06%	-2.91%	-5.33%
	σ%	-1.16%	-2.79%	-4.01%
PostgreSQL	t_m%	-0.61%	-0.62%	-0.10%
	σ%	+0.39%	-2.61%	-1.02%

Fonte: Produzido pelos autores.

Para observar a distribuição dos dados, a Figura 5.10 apresenta as medições referentes à ordenação crescente na máquina 2xlarge. As Figuras A.51, A.52, A.53, A.54 e A.55, dispostas no Apêndice A, mostram os resultados dos outros cenários de teste.

Tal qual nos testes de inserção única, o MariaDB apresentou os melhores resultados em todos os casos de teste, e o MySQL apresentou os piores.

Diferente do teste de inserção única, não houve uma mudança percentual extremamente alta na troca de máquinas. O Microsoft SQL se manteve como o que mais sofreu variação do desvio padrão nas trocas, mas nesse cenário foi o PostgreSQL que sofreu menor variação do tempo de resposta em função do hardware.

Além disso, MariaDB foi o único banco de dados que não experimentou diminuição do tempo médio de resposta na troca de hardware. Os outros três apresentaram melhoria em todas as ordenações, se considerarmos a mudança percentual do MySQL no caso crescente como desprezível, dado seu valor de diferença ínfimo.

Na Tabela 5.8 está uma comparação do tempo de inserção única triplicado com o tempo de inserção em rajadas de 3 imagens. Em azul está destacado o menor tempo dentre os dois comparados.

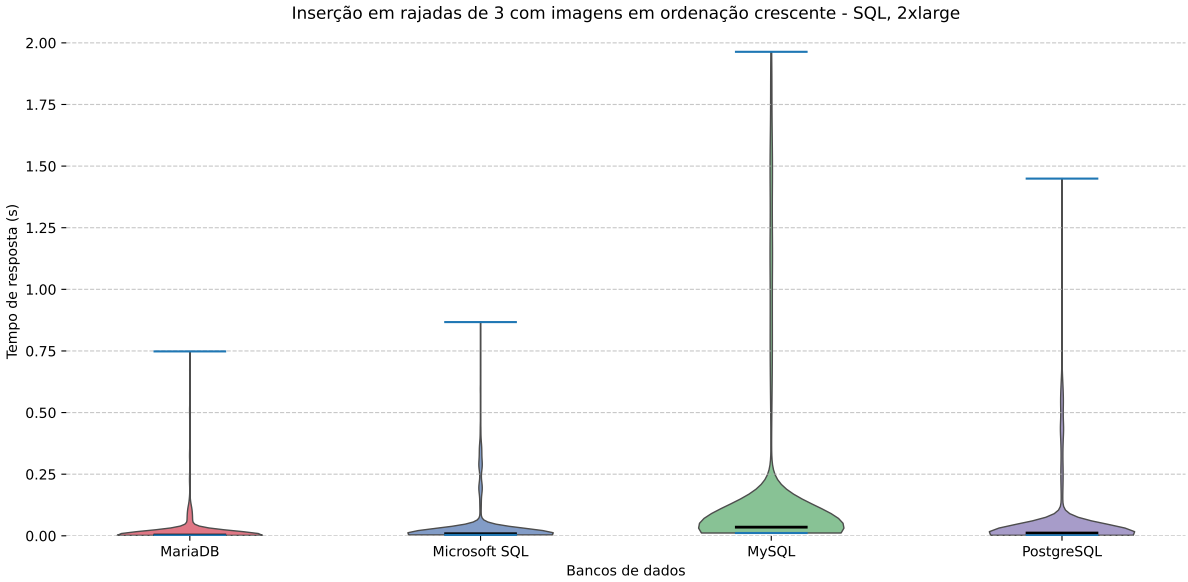

Figura 5.10 – Gráfico dos tempos de resposta (inserção em rajadas, crescente, SQL, 2xlarge)

Fonte: Produzido pelos autores.

Para quase todos os casos analisados, a inserção por rajadas apresentou um tempo de resposta melhor, exceto para o Microsoft SQL, que em todas as ordenações e máquinas, teve um tempo de resposta das rajadas pior, e a diferença de tempo não foi desprezível. Isso indica que o Microsoft SQL apresenta melhor performance ao trabalhar com inserções únicas seguidas ao invés de inserções em rajada.

Tabela 5.8 – Comparação do resultado do teste de inserção única triplicado com o teste de rajadas

Banco de dados e resultados		Ordenação de imagens e máquina utilizada					
		Crescente		Decrescente		Aleatório	
		2xlarge	8xlarge	2xlarge	8xlarge	2xlarge	8xlarge
MariaDB	$t_{m,singular} \times 3$ (s)	0.019293	0.019218	0.019983	0.020088	0.018780	0.018591
	$t_{m,rajada}$ (s)	0.017987	0.018219	0.018087	0.018188	0.018375	0.018746
Microsoft SQL	$t_{m,singular} \times 3$ (s)	0.027618	0.026211	0.026379	0.027117	0.027764	0.026406
	$t_{m,rajada}$ (s)	0.033539	0.032120	0.032164	0.030669	0.033131	0.032420
MySQL	$t_{m,singular} \times 3$ (s)	0.158643	0.147393	0.152466	0.155133	0.153870	0.159321
	$t_{m,rajada}$ (s)	0.136303	0.136382	0.135565	0.131621	0.142807	0.135192
PostgreSQL	$t_{m,singular} \times 3$ (s)	0.057549	0.054849	0.060747	0.054441	0.057189	0.055683
	$t_{m,rajada}$ (s)	0.051306	0.050993	0.051113	0.050797	0.051772	0.051722

Fonte: Produzido pelos autores.

Avaliando a Tabela 5.9, percebemos alguns resultados interessantes. Os testes na ordenação aleatória foram o teste com menor desvio padrão para todos os bancos de dados, e os testes em ordenação crescente os com maior desvio padrão.

Houve também uma predominância dos testes aleatórios para o pior tempo de resposta médio, e dos testes decrescentes para o melhor tempo médio. Comparando com os

resultados analisados no teste de inserção única, os testes de rajada apresentaram maior homogeneidade nas categorias analisadas.

Tabela 5.9 – Casos com pior e melhor resultado para cada banco de dados (inserção em rajada, SQL)

BD	Pior tempo médio	Melhor tempo médio	Menor distribuição	Maior distribuição
MariaDB	Aleatório - 8x	Crescente - 2x	Aleatório - 8x	Crescente - 8x
Microsoft SQL	Crescente - 2x	Decrescente - 8x	Aleatório - 2x	Crescente - 2x
MySQL	Aleatório - 2x	Decrescente - 8x	Aleatório - 8x	Crescente - 2x
PostgreSQL	Aleatório - 2x	Decrescente - 8x	Aleatório - 8x	Crescente - 8x

Fonte: Produzido pelos autores.

De posse do menor e maior valor de tempo de resposta para cada caso, foram elaborados os gráficos das Figuras 5.11 e 5.12.

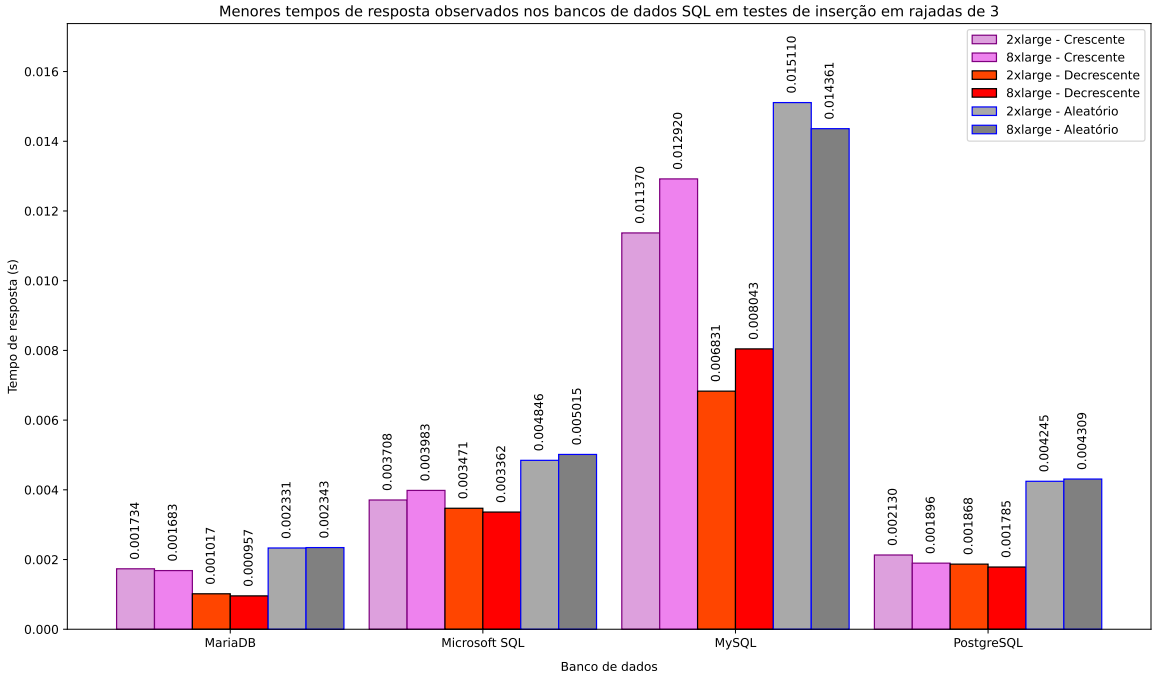

Figura 5.11 – Gráfico do menor tempo de resposta de inserção em rajadas em SQL

Fonte: Produzido pelos autores.

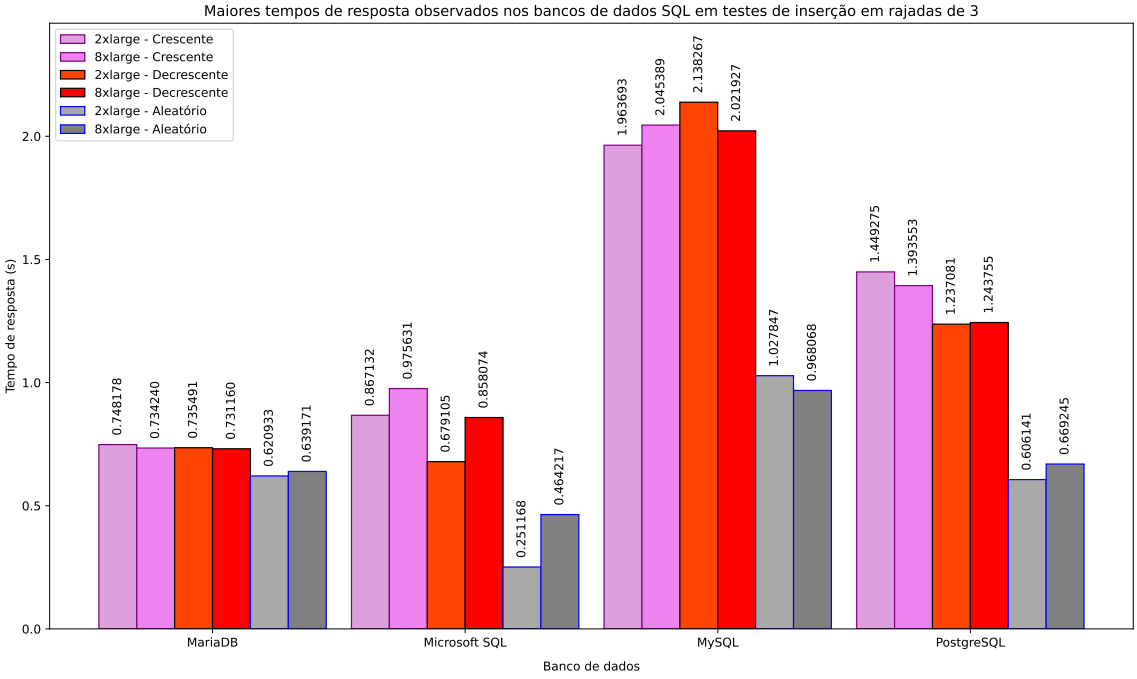

Figura 5.12 – Gráfico do maior tempo de resposta de inserção em rajadas em SQL

Fonte: Produzido pelos autores.

Contrastando com o caso de inserção única, o MySQL apresentou os maiores picos em ambos os casos.

A flutuação de casos em que a troca para a máquina com hardware superior resultou em tempos de resposta menores não nos permite traçar uma correspondência direta entre o *hardware* e os valores mínimo e máximo atingidos pelos BDs.

5.2.3 Testes de deleção

Para os testes de deleção, utilizamos o ID para identificar a entrada a ser deletada, para que o tempo de resposta obtido representasse o mais fielmente possível o tempo necessário para o banco de dados apagar o arquivo binário no registro.

A Tabela 5.10 mostra os resultados obtidos, e a Tabela 5.11 demonstra as mudanças percentuais na troca de máquinas.

Tabela 5.10 – Resultados dos testes de deleção dos bancos de dados relacionais

Banco de dados e resultados		Ordenação de imagens e máquina utilizada					
		Crescente		Decrescente		Aleatório	
		2xlarge	8xlarge	2xlarge	8xlarge	2xlarge	8xlarge
MariaDB	t_m (s)	0.018452	0.017525	0.019377	0.039572	0.021552	0.034812
	σ (s)	0.045664	0.043723	0.047006	0.107704	0.056148	0.091309
Microsoft SQL	t_m (s)	0.002891	0.002704	0.002887	0.003295	0.002892	0.003581
	σ (s)	0.001094	0.001034	0.001097	0.001409	0.001943	0.002340
MySQL	t_m (s)	0.029757	0.025019	0.029547	0.035693	0.032793	0.023786
	σ (s)	0.086794	0.075332	0.087612	0.103857	0.098313	0.060701
PostgreSQL	t_m (s)	0.008499	0.008282	0.007675	0.008977	0.009229	0.008160
	σ (s)	0.034665	0.034942	0.035644	0.043341	0.034852	0.030106

Fonte: Produzido pelos autores.

Tabela 5.11 – Mudança percentual dos resultados ao trocar da 2xlarge para a 8xlarge (deleção, SQL)

Banco de dados e mudança percentual		Ordenação de imagens		
		Crescente	Decrescente	Aleatório
MariaDB	t_m%	-5.02%	+104.22%	+61.53%
	σ%	-4.25%	+129.13%	+62.62%
Microsoft SQL	t_m%	-6.47%	+14.13%	+23.82%
	σ%	-5.48%	+28.44%	+20.43%
MySQL	t_m%	-15.92%	+20.80%	-27.47%
	σ%	-13.21%	+18.54%	-38.26%
PostgreSQL	t_m%	-2.55%	+16.96%	-11.58%
	σ%	+0.80%	+21.59%	-13.62%

Fonte: Produzido pelos autores.

Dada a discrepância dos aumentos observados no MariaDB, é importante considerar que podem ter ocorrido interferências no sistema que alteraram o resultado final.

As Figuras 5.13 e 5.14 apresentam, respectivamente, a distribuição dos tempos de resposta no caso de ordenação ascendente na máquina 2xlarge e o *zoom* para observar a curva do Microsoft SQL, que apresentou resultados muito menores que os outros BDs.

As distribuições dos outros cenários de teste estão dispostas no Apêndice A, representadas pelas Figuras A.56, A.57, A.58, A.59, A.60, A.61, A.62, A.63, A.64 e A.65,

É um problema comum o fato de operações de deleção em SQL tomarem muito tempo, ao ponto de existirem diversas alternativas para o comando DELETE que se provam mais eficazes, como é abordado em uma publicação de Chris Saxon (2023). Dessa forma, os resultados do Microsoft SQL, que apresentou o melhor resultado tanto em tempo de resposta médio quanto na distribuição das medidas, indicam a presença de uma otimização interna do banco de dados para as operações de deleção.

O PostgreSQL ficou em segundo lugar em ambos os parâmetros avaliados, fato esse que não foi observado nos testes anteriores. Esse resultado demonstra o melhor desempenho do PostgreSQL para operações de deleção.

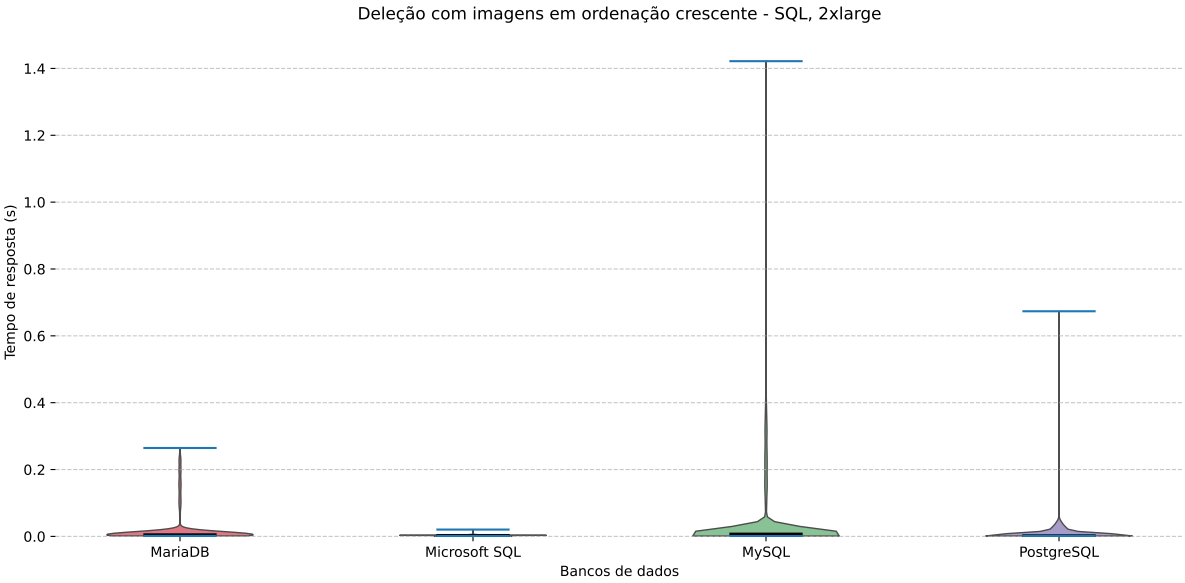

Figura 5.13 – Gráfico dos tempos de resposta (deleção, crescente, SQL, 2xlarge)

Fonte: Produzido pelos autores.

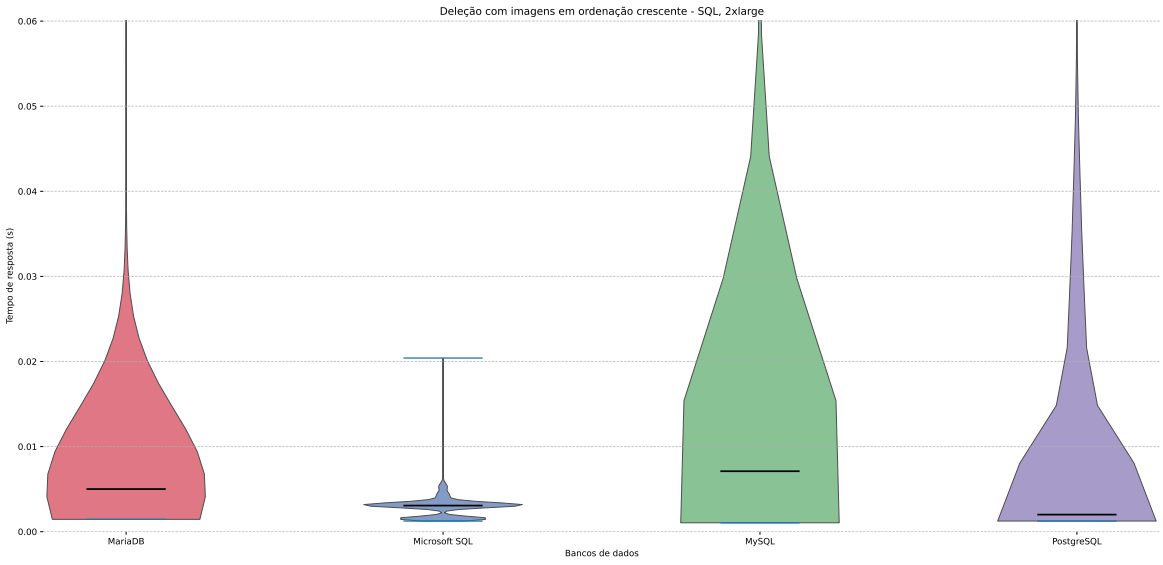

Figura 5.14 – *Zoom* para inspeção da curva do Microsoft SQL da Figura 5.17

Fonte: Produzido pelos autores.

Avaliando a Tabela 5.12, percebemos uma tendência dos piores resultados de tempo médio e de distribuição ocorrerem exatamente no mesmo ambiente, o mesmo sendo válido para os melhores resultados. Além disso, os testes em ordenação crescente apareceram nos melhores resultados metade das vezes, e não figuraram dentre os piores resultados nenhuma vez.

Tabela 5.12 – Casos com pior e melhor resultado para cada banco de dados (deleção, SQL)

BD	Pior tempo médio	Melhor tempo médio	Menor distribuição	Maior distribuição
MariaDB	Decrescente - 8x	Crescente - 8x	Crescente - 8x	Decrescente - 8x
Microsoft SQL	Aleatório - 8x	Crescente - 8x	Crescente - 8x	Aleatório - 8x
MySQL	Decrescente - 8x	Aleatório - 8x	Aleatório - 8x	Decrescente - 8x
PostgreSQL	Aleatório - 2x	Decrescente - 2x	Aleatório - 8x	Decrescente - 8x

Fonte: Produzido pelos autores.

As Figuras 5.15 e 5.16 mostram a relação dos menores e maiores valores do tempo de resposta para cada banco de dados.

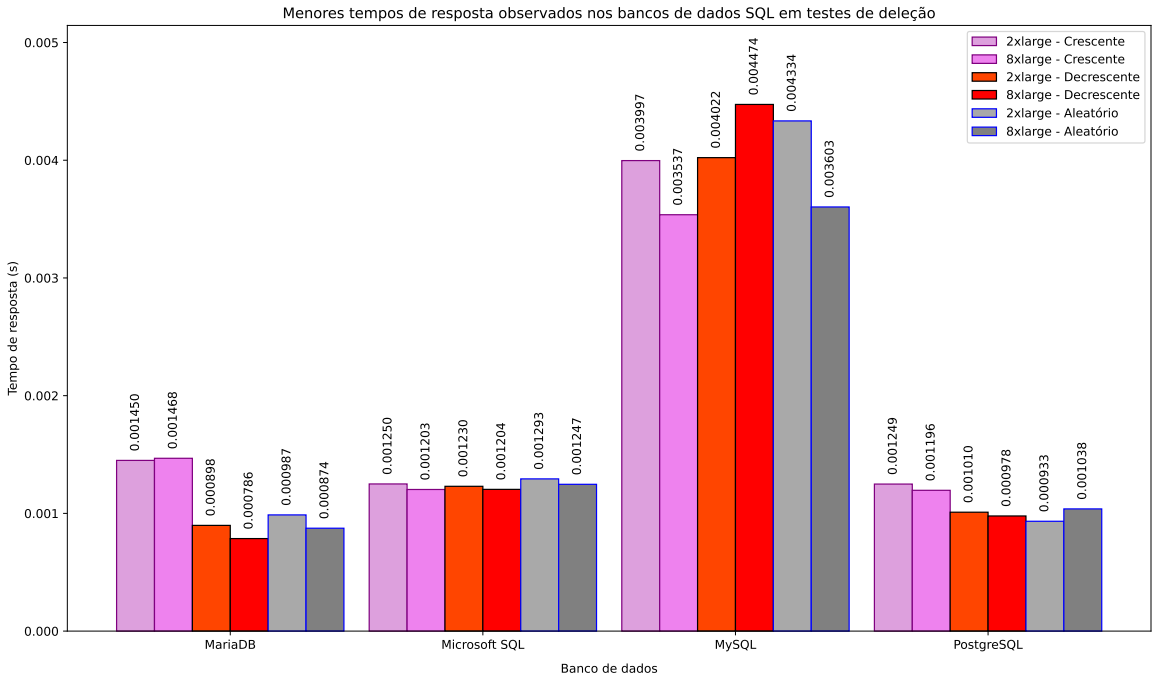

Figura 5.15 – Gráfico do menor tempo de resposta de deleção em SQL

Fonte: Produzido pelos autores.

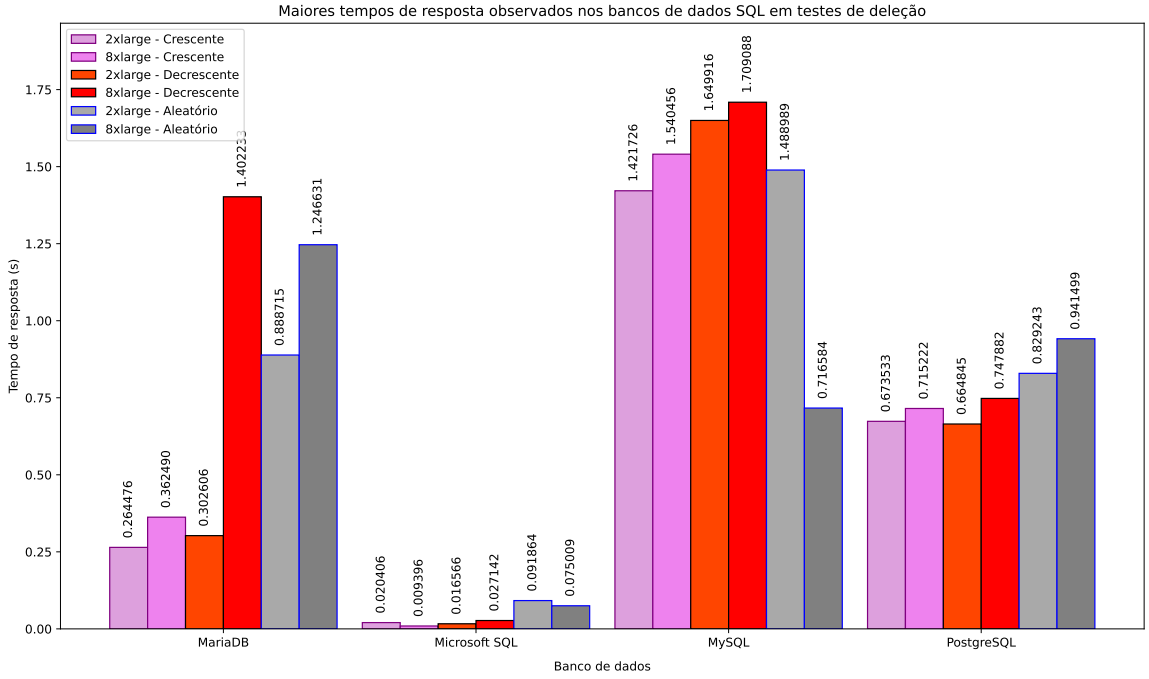

Figura 5.16 – Gráfico do maior tempo de resposta de deleção em SQL

Fonte: Produzido pelos autores.

De maneira geral, o menor valor do tempo de resposta diminui com a troca para a máquina de *hardware* superior, mas o mesmo já não pode ser dito a respeito do maior tempo de resposta. Porém, é bom lembrar que esse último dado pode sofrer considerável flutuação em função de interferências do ambiente que não podem ser controladas.

5.2.4 Testes de leitura

Para os testes de leitura, realizados com seleção pelo número identidade associado à cada imagem, foram utilizados os bancos de dados apenas na ordenação crescente. A leitura das imagens não apresentou variações expressivas entre as ordenações quando feita pelo número identidade, sendo necessário apenas uma das ordenações. A Tabela 5.13 mostra os resultados obtidos, bem como a mudança percentual ao trocar de máquina. Em azul estão destacados os melhores resultados, e em vermelho os piores.

Tabela 5.13 – Resultados dos testes de leitura dos bancos de dados relacionais

Banco de dados e resultados		Máquina utilizada e mudança percentual		
		2xlarge	8xlarge	%
MariaDB	t_m (s)	0.017739	0.015741	-11.26%
	σ (s)	0.048105	0.041096	-14.57%
Microsoft SQL	t_m (s)	0.001701	0.001940	+14.05%
	σ (s)	0.003874	0.004127	+6.53%
MySQL	t_m (s)	0.000878	0.000905	+3.08%
	σ (s)	0.001644	0.001729	+5.17%
PostgreSQL	t_m (s)	0.003361	0.003517	+4.64%
	σ (s)	0.008703	0.009154	+5.18%

Fonte: Produzido pelos autores.

A distribuição das medidas pode ser observada nas Figuras 5.17 e 5.19. Devido à escala da distribuição do MariaDB, as Figuras 5.18 e 5.20 representam um *zoom* para inspeção da curva dos outros bancos de dados.

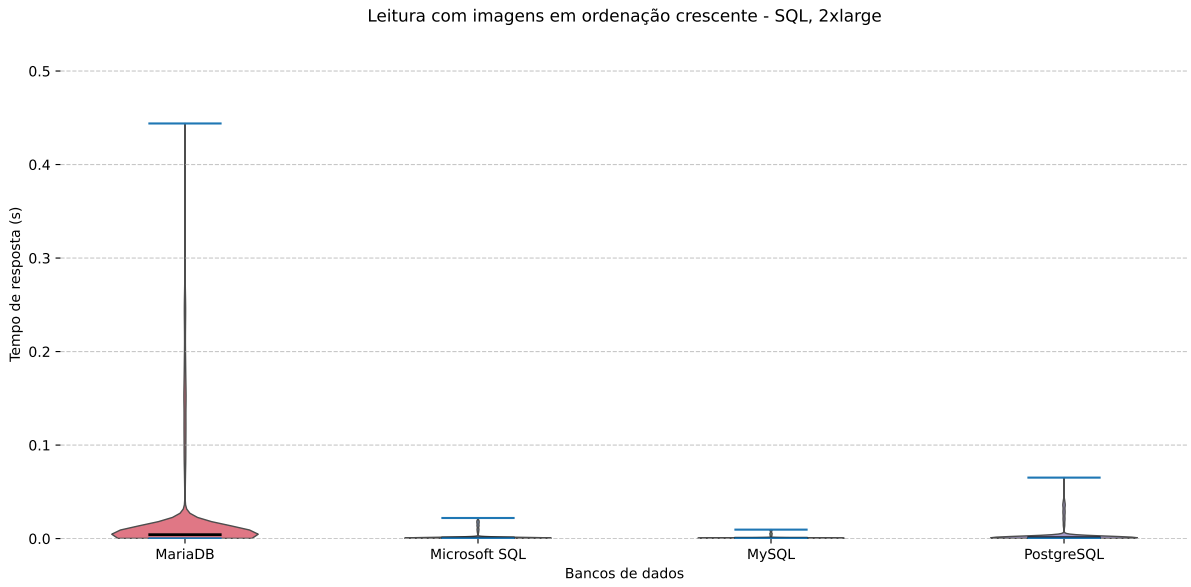

Figura 5.17 – Gráfico dos tempos de resposta (leitura, SQL, 2xlarge)

Fonte: Produzido pelos autores.

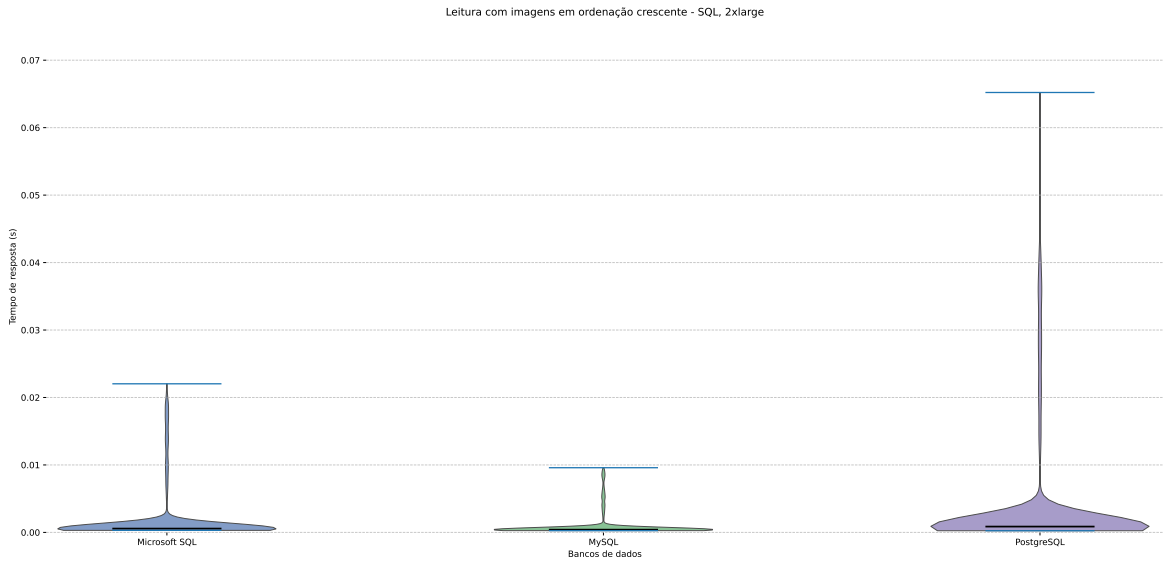

Figura 5.18 – *Zoom* para inspeção das curvas do Microsoft SQL, MySQL e PostgreSQL da Figura 5.17

Fonte: Produzido pelos autores.

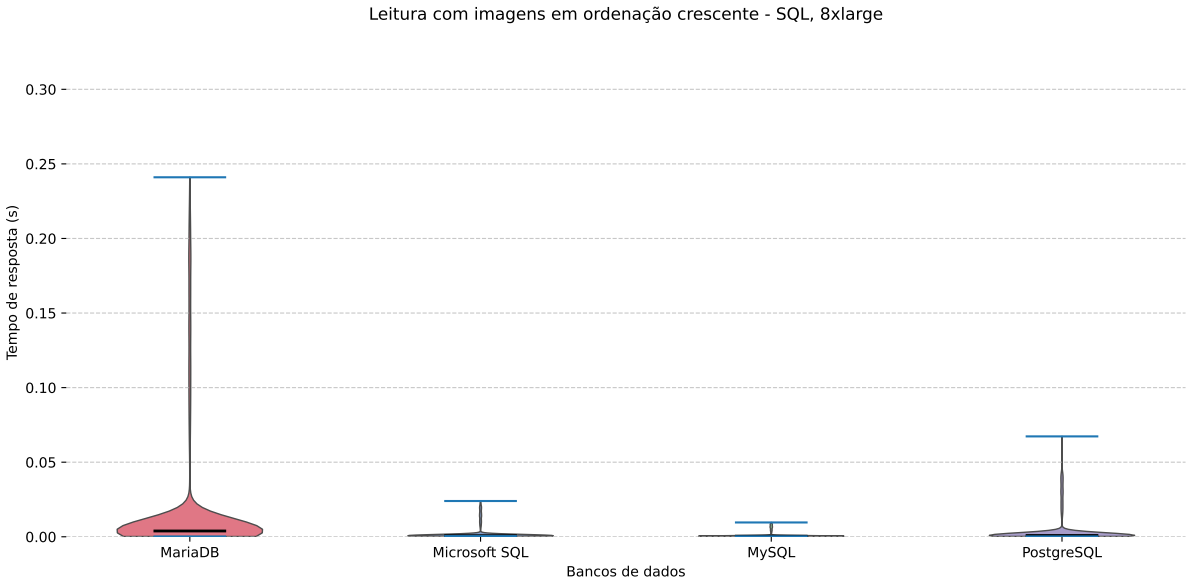

Figura 5.19 – Gráfico dos tempos de resposta (leitura, SQL, 8xlarge)

Fonte: Produzido pelos autores.

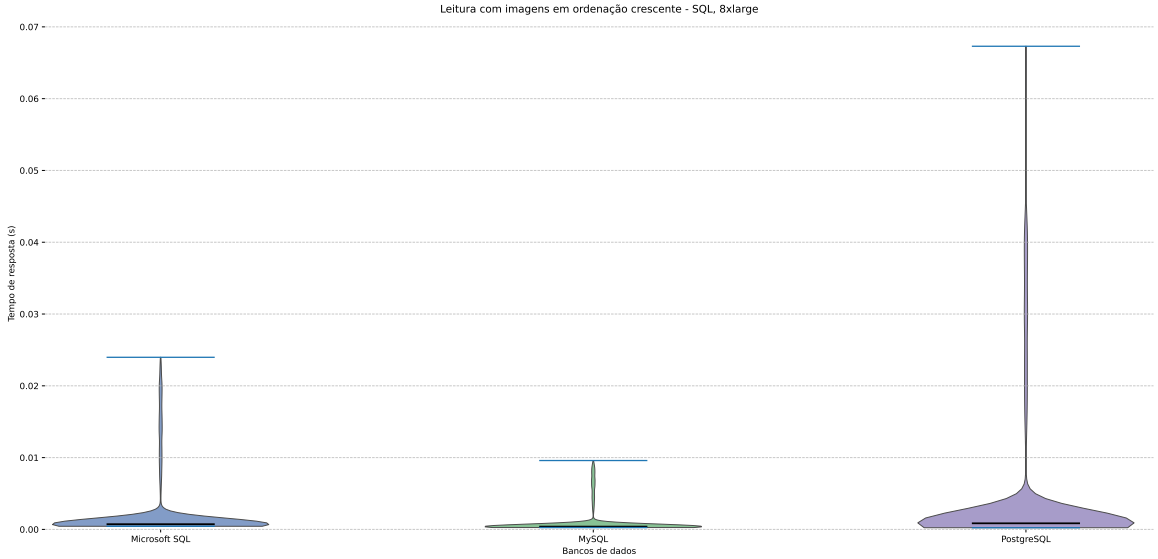

Figura 5.20 – *Zoom* para inspeção das curvas do Microsoft SQL, MySQL e PostgreSQL da Figura 5.19

Fonte: Produzido pelos autores.

Por fim, as Figuras 5.21 e 5.22 mostram a relação dos menores e maiores valores do tempo de resposta para cada banco de dados.

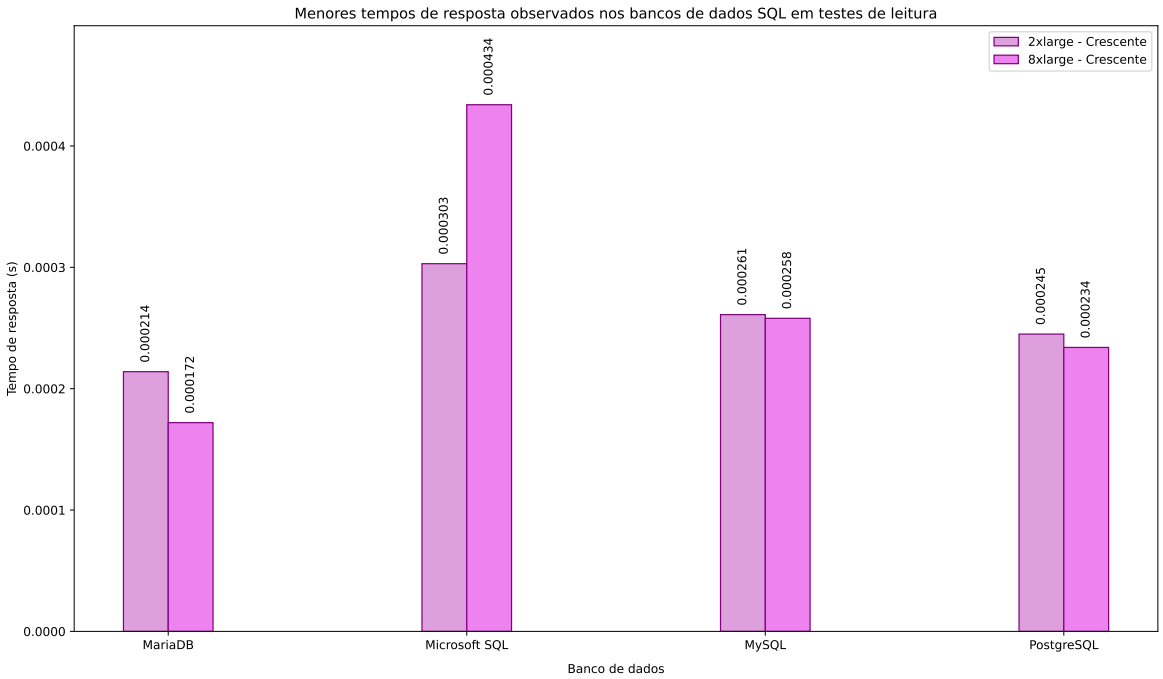

Figura 5.21 – Gráfico do menor tempo de resposta de leitura em SQL

Fonte: Produzido pelos autores.

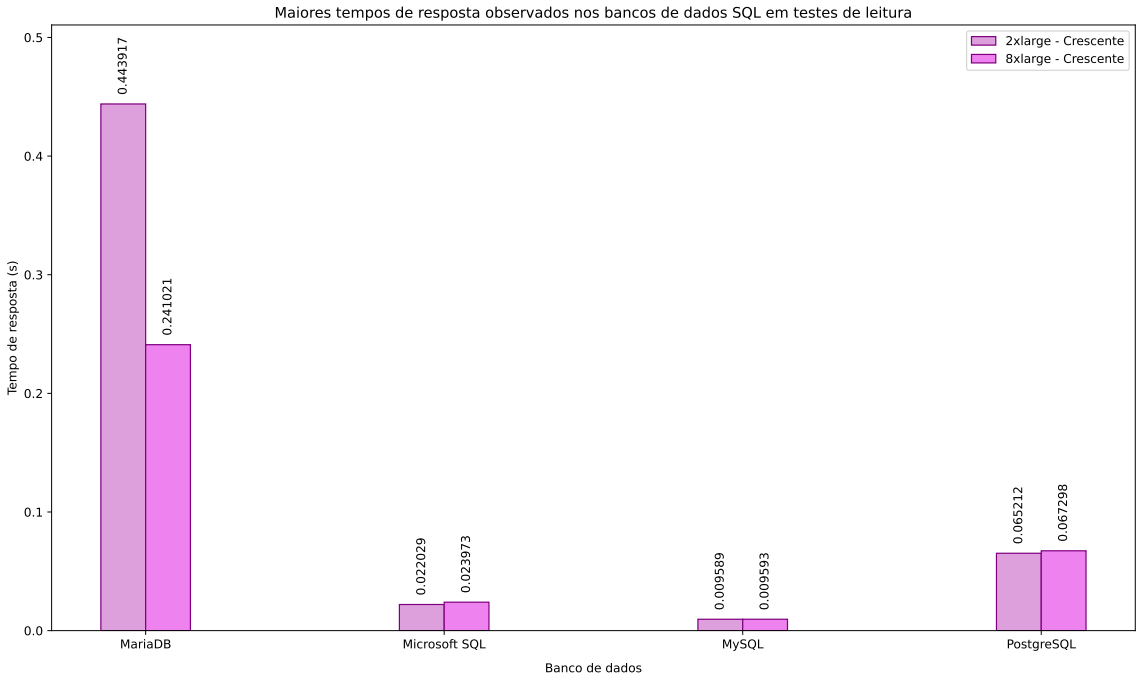

Figura 5.22 – Gráfico do maior tempo de resposta de leitura em SQL

Fonte: Produzido pelos autores.

O teste de leitura teve resultados bem diferentes se comparados aos testes anteriores. O MariaDB apresentou os piores resultados de tempo médio e distribuição, ao passo que o MySQL, até então demonstrando os piores resultados nos outros testes, teve o melhor desempenho nos dois critérios avaliados.

Não só isso, mas o MariaDB foi o único a apresentar melhoria dos resultados ao trocar para a máquina 8xlarge, tendo sido também o mais impactado pela troca, seguido do Microsoft SQL.

5.3 Bancos de dados NoSQL

Tal qual para os bancos de dados SQL, analisamos o tempo médio da operação em segundos (t_m) e o desvio padrão (σ) e os valores de tempo de execução mínimo e máximo de cada conjunto de operações.

5.3.1 Testes de inserção única

A Tabela 5.14 mostra os resultados obtidos, destacando em azul os melhores resultados por ordenação e em vermelho os piores, e a Tabela 5.15 dispõe a mudança percentual das medidas feitas na máquina p3.8xlarge em relação à p3.2xlarge.

Tabela 5.14 – Resultados dos testes de inserção única dos bancos de dados não relacionais

Banco de dados e resultados		Ordenação de imagens e máquina utilizada					
		Crescente		Decrescente		Aleatório	
		2xlarge	8xlarge	2xlarge	8xlarge	2xlarge	8xlarge
Cassandra	t_m (s)	0.033033	0.032038	0.034661	0.030013	0.031423	0.028062
	σ (s)	0.020892	0.020695	0.020995	0.021363	0.022239	0.022553
MongoDB	t_m (s)	0.007561	0.004864	0.005179	0.002643	0.004894	0.004738
	σ (s)	0.087443	0.048348	0.027425	0.014837	0.053302	0.046983
Redis	t_m (s)	0.000551	0.000516	0.000514	0.000489	0.000490	0.000444
	σ (s)	0.001099	0.001054	0.000875	0.000872	0.001000	0.000937
Solr	t_m (s)	0.250427	0.249445	0.245082	0.244070	0.255253	0.252655
	σ (s)	0.443306	0.426109	0.430390	0.421872	0.419753	0.408760

Fonte: Produzido pelos autores.

Tabela 5.15 – Mudança percentual dos resultados ao trocar da 2xlarge para a 8xlarge (inserção única, NoSQL)

Banco de dados e mudança percentual		Ordenação de imagens		
		Crescente	Decrescente	Aleatório
Cassandra	t_m%	-3.01%	-13.41%	-10.70%
	σ%	-0.94%	+1.75%	+1.41%
MongoDB	t_m%	-35.67%	-48.97%	-3.19%
	σ%	-44.71%	-45.90%	-11.86%
Redis	t_m%	-6.35%	-4.86%	-9.39%
	σ%	-4.09%	-0.34%	-6.30%
Solr	t_m%	-0.39%	-0.41%	-1.02%
	σ%	-3.88%	-1.98%	-2.62%

Fonte: Produzido pelos autores.

Facilmente vemos o destaque do Redis em todos os parâmetros, pela Tabela 5.14, obtendo resultados significativamente menores. A tabela Tabela 5.15 mostra consistente redução dos tempos médios na troca de máquinas, com o desvio padrão também reduzindo na maior parte dos casos.

A Figura 5.23 apresenta a distribuição das medidas obtidas para a ordenação crescente na máquina 2xlarge.

Uma vez que as medições do Solr se demonstraram muito maiores em relação aos outros bancos de dados, especialmente comparando ao Redis, foi incluído também, na Figura 5.24, uma visão ampliada das curvas do Cassandra e Redis, removendo o Solr e deixando em vista apenas uma parte da curva do MongoDB, de maneira a preservar a visualização dos dados de menor escala.

Os resultados das outras ordenações e máquinas estão disponíveis no Apêndice A, representandos nas Figuras A.66, A.67, A.68, A.69, A.70, A.71, A.72, A.73, A.74 e A.75

O Redis apresentou os melhores resultados em todas as categorias, de maneira nítida. O Solr, por sua vez, teve o pior desempenho para ambas as métricas, e seus resultados estão muito distantes do outros bancos de dados.

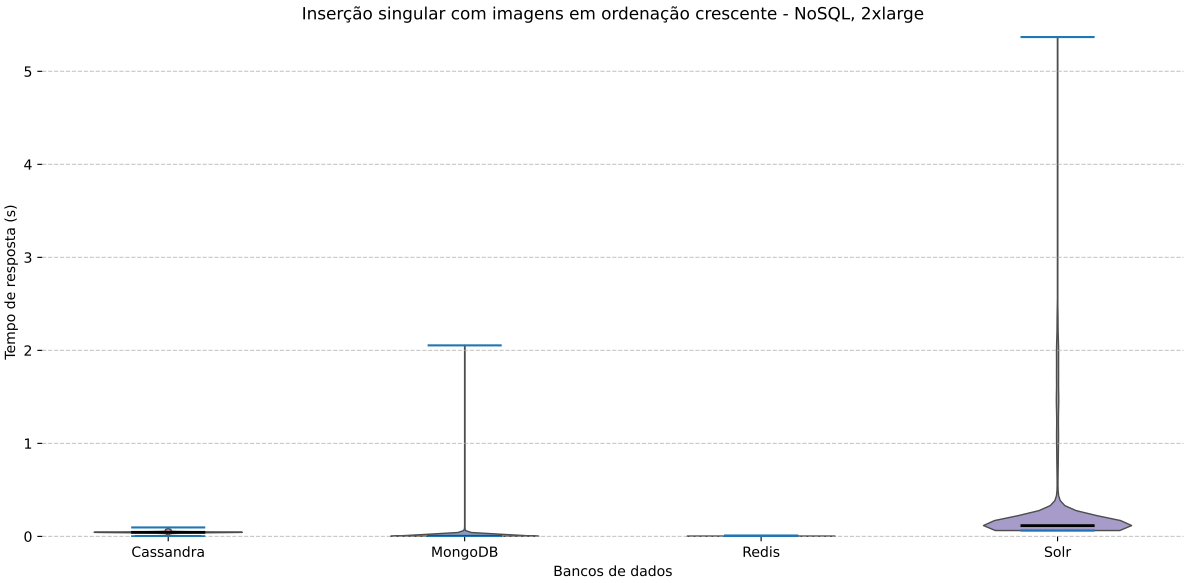

Figura 5.23 – Gráfico dos tempos de resposta (inserção única, crescente, NoSQL, 2xlarge)

Fonte: Produzido pelos autores.

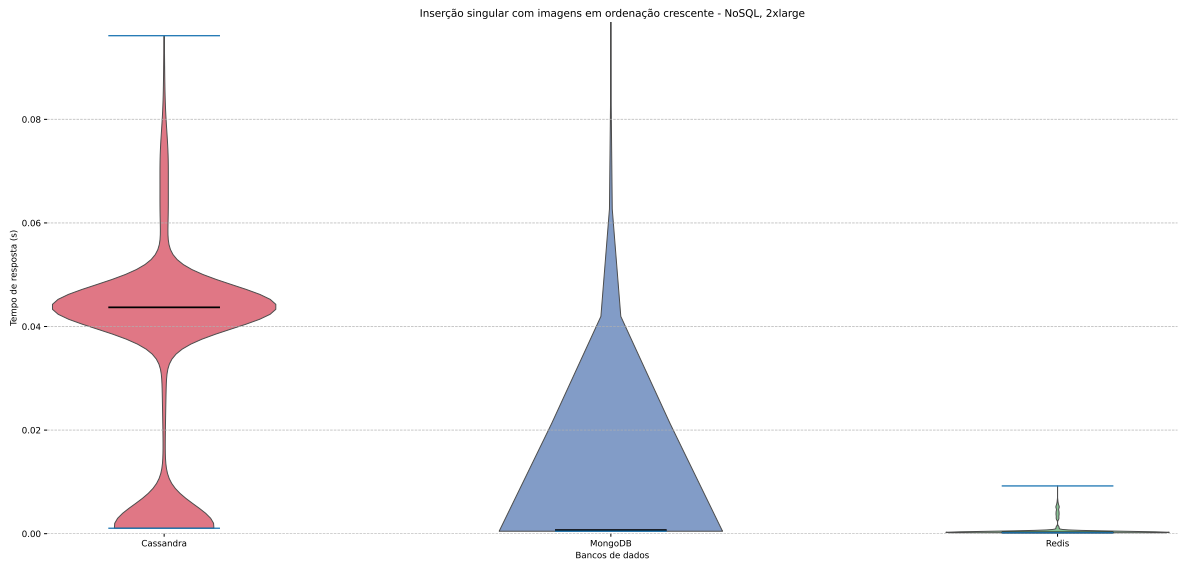

Figura 5.24 – *Zoom* para inspeção das curvas do Cassandra e Redis da Figura 5.23

Fonte: Produzido pelos autores.

Em quase todos os casos, a performance dos bancos de dados escolhidos melhorou na troca para a máquina com mais recursos computacionais, notavelmente o MongoDB, que aparenta conseguir utilizar recursos extras de forma mais agressiva, com mais instabilidade entre *hardwares*. Apenas o Cassandra teve um aumento percentual no desvio padrão com a troca de máquinas.

O Solr se manteve o mais percentualmente consistente na troca de *hardware*, o que não é algo surpreendente, dados os elevados tempos de resposta que apresentou. Tamanha lentidão para a inserção de arquivos binários, mesmo os de tamanho reduzido, indicam que o Solr pode não ser a melhor escolha para tratamento de arquivos binários.

O MongoDB se mostrou consistente em relação ao tempo de armazenamento necessário comparado ao tamanho do binário sendo armazenado, de forma bem próxima ao linear, enquanto o Cassandra agrupa o tempo de suas requisições de forma a se manter o mais estável possível dentro do tempo esperado no contato com o *cluster*. Esse comportamento, porém, pode ser causado por uma dependência do conector de Python utilizado para o acesso.

Tabela 5.16 – Casos com pior e melhor resultado para cada banco de dados (inserção única, NoSQL)

BD	Pior tempo médio	Melhor tempo médio	Menor distribuição	Maior distribuição
Cassandra	Decrescente - 2x	Aleatório - 8x	Crescente - 8x	Aleatório - 8x
MongoDB	Crescente - 2x	Decrescente - 8x	Decrescente - 8x	Crescente - 2x
Redis	Crescente - 2x	Aleatório - 8x	Decrescente - 8x	Crescente - 2x
Solr	Aleatório - 2x	Decrescente - 8x	Aleatório - 8x	Crescente - 2x

Fonte: Produzido pelos autores.

Avaliando a Tabela 5.16, vemos alguns fatores repetidos, principalmente na consistência de melhores performances em máquinas mais poderosas pelos bancos de dados NoSQL. Não apareceu consistência entre os piores e melhores tempos médios além do *hardware* utilizado, e as diferenças reais entre os tempos observados foram pequenas, já que na maior parte dos casos, os NoSQLs possuem execuções bem rápidas.

De posse do menor e maior valor de tempo de resposta para cada caso, foram elaborados os gráficos das Figuras 5.25 e 5.26.

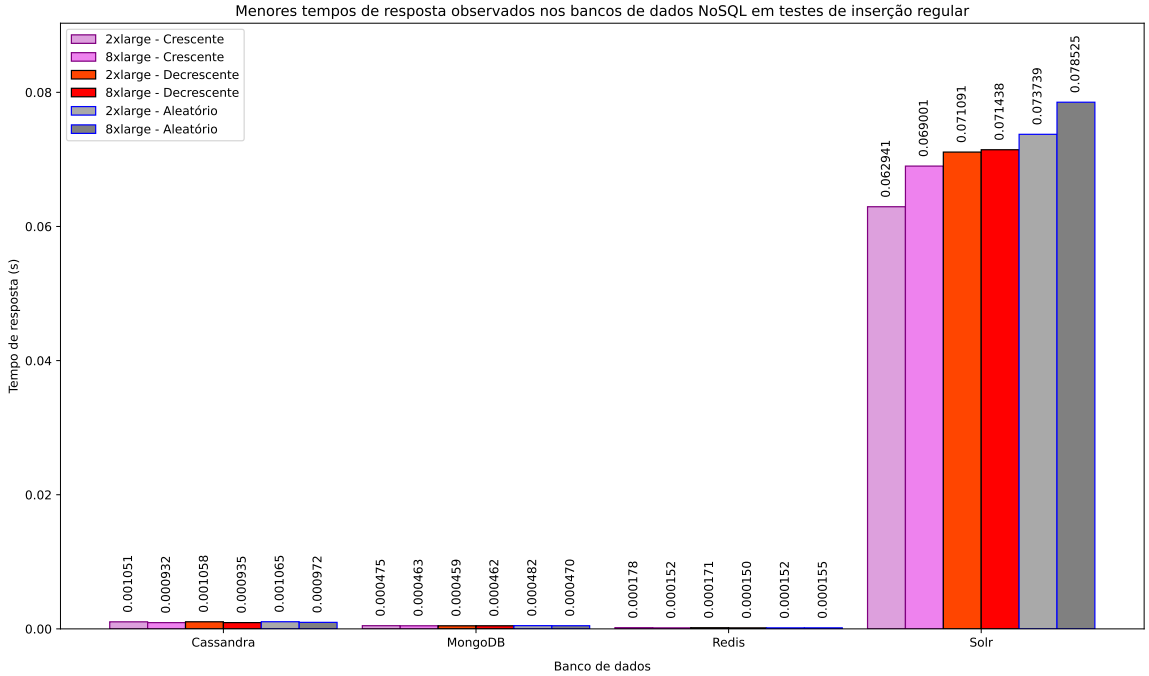

Figura 5.25 – Gráfico do menor tempo de resposta de inserção única em NoSQL

Fonte: Produzido pelos autores.

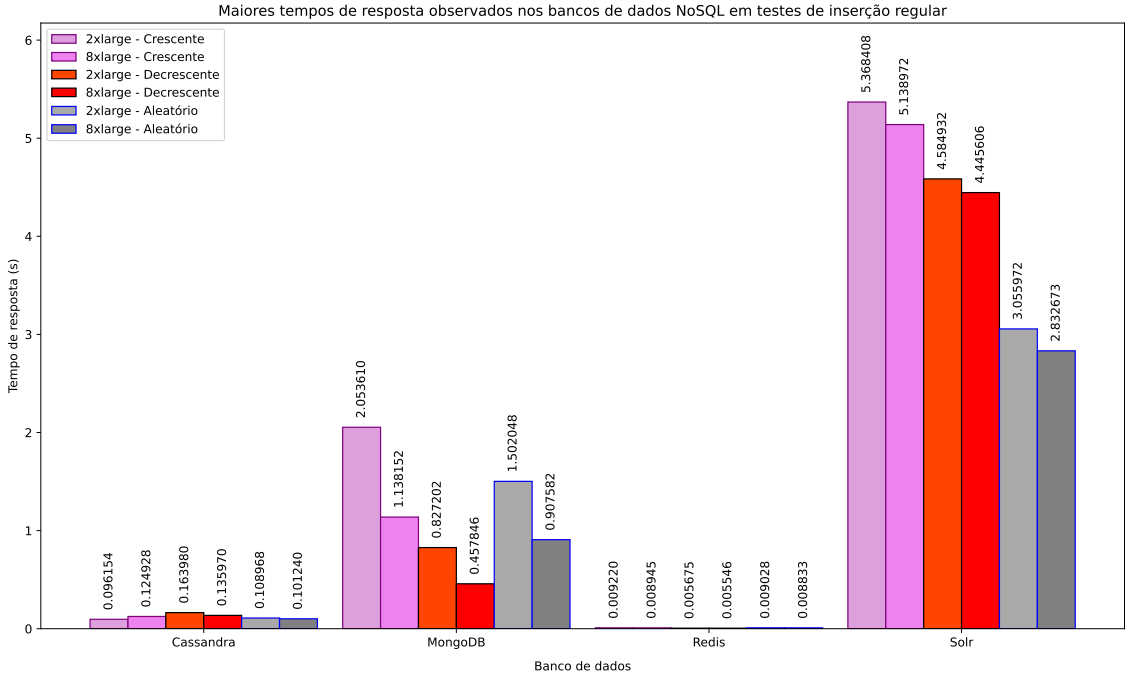

Figura 5.26 – Gráfico do maior tempo de resposta de inserção única em NoSQL

Fonte: Produzido pelos autores.

Temos o Redis como melhor performance tanto no maior e menor tempo de resposta em inserção única. O MongoDB flutuou entre respostas muito rápidas e respostas mais

lentas, enquanto o Cassandra se mostrou extremamente consistente.

Para o Solr, mesmo os menores tempo de resposta se demonstraram muito além dos maiores tempos dos outros bancos de dados, e seus maiores valores para todos os casos foram extremamente altos.

A máquina de *hardware* superior conseguiu causar uma ligeira diminuição dos tempos máximos de resposta em quase todos os casos, e seu impacto foi ainda menor para as medições de menor tempo de resposta, chegando até mesmo a apresentar um aumento de tal medida para o Solr.

5.3.2 Testes de inserção em rajadas de 3 imagens

A Tabela 5.17 mostra os resultados obtidos para os casos testados, destacando em azul o melhor resultado por ordenação e em vermelho o pior, e a Tabela 5.18 dispõe a mudança percentual das medidas feitas em cada máquina.

O fato do Cassandra possuir uma configuração avançada editável em seu .yaml chamada $batch\_size\_fail\_threshold\_in\_kb$, que é específica para envios em *batch* - que utilizamos para os testes de inserção em rajadas - e cujo valor padrão é 50KiB é um forte indicador de que o envio de imagens pesadas em rajada é contraindicado para o Cassandra. De acordo com a documentação publicada no DataStax (2023a), aumentar esse limite pode resultar em instabilidade.

De maneira a testar esse cenário, alteramos essa configuração para 128000KiB, um valor que comportasse um burst bem maior que um grupo com as 3 maiores imagens do grupo de testes, mas isso resultou em um *timeout*.

Como nas ordenações crescente e decrescente as imagens mais pesadas estão juntas, resultando em um *batch* de tamanho elevado, essas ordenações foram puladas para os testes de rajadas do Cassandra. Como a densidade de imagens de tamanho pequeno é maior, a ordenação aleatória foi finalizado com sucesso.

Tabela 5.17 – Resultados dos testes de inserção em rajadas dos bancos de dados não relacionais

Banco de dados e resultados		Ordenação de imagens e máquina utilizada					
		Crescente		Decrescente		Aleatório	
		2xlarge	8xlarge	2xlarge	8xlarge	2xlarge	8xlarge
Cassandra	t_m (s)	-	-	-	-	0.047880	0.046000
	σ (s)	-	-	-	-	0.024650	0.024563
MongoDB	t_m (s)	0.019739	0.011138	0.018498	0.015976	0.017421	0.015521
	σ (s)	0.068232	0.062482	0.058863	0.101269	0.124579	0.106045
Redis	t_m (s)	0.000896	0.000905	0.000883	0.000758	0.001031	0.000976
	σ (s)	0.001880	0.001902	0.001768	0.001331	0.001495	0.001442
Solr	t_m (s)	0.573098	0.532842	0.581296	0.531759	0.584428	0.526558
	σ (s)	1.255838	1.204621	1.285071	1.209080	0.721226	0.667023

Fonte: Produzido pelos autores.

Tabela 5.18 – Mudança percentual dos resultados ao trocar da 2xlarge para a 8xlarge (inserção em rajadas, NoSQL)

Banco de dados e mudança percentual		Ordenação de imagens		
		Crescente	Decrescente	Aleatório
Cassandra	t_m%	-	-	-3.93%
	σ%	-	-	-0.35%
MongoDB	t_m%	-43.57%	-13.63%	-10.91%
	σ%	-8.43%	+72.04%	-14.88%
Redis	t_m%	+1.00%	-14.16%	-5.33%
	σ%	+1.17%	-24.72%	-3.55%
Solr	t_m%	-7.02%	-8.52%	-9.90%
	σ%	-4.08%	-5.91%	-7.52%

Fonte: Produzido pelos autores.

A distribuição das medidas obtidas para inserção em rajadas em ordenação aleatória na máquina 2x está representada na Figura 5.27.

Como as medições do Solr novamente se demonstraram muito maiores em relação aos outros bancos de dados, especialmente se comparado ao Redis, a Figura 5.28 apresenta uma visão em *zoom* das curvas do Cassandra, Mongo e Redis, removendo o Solr para preservar a visualização de todos os dados.

Os resultados dos outros casos de ordenação e máquinas estão dispostos no Apêndice A, representados pelas Figuras A.76, A.77, A.78, A.79, A.80, A.81, A.82, A.83, A.84 e A.85.

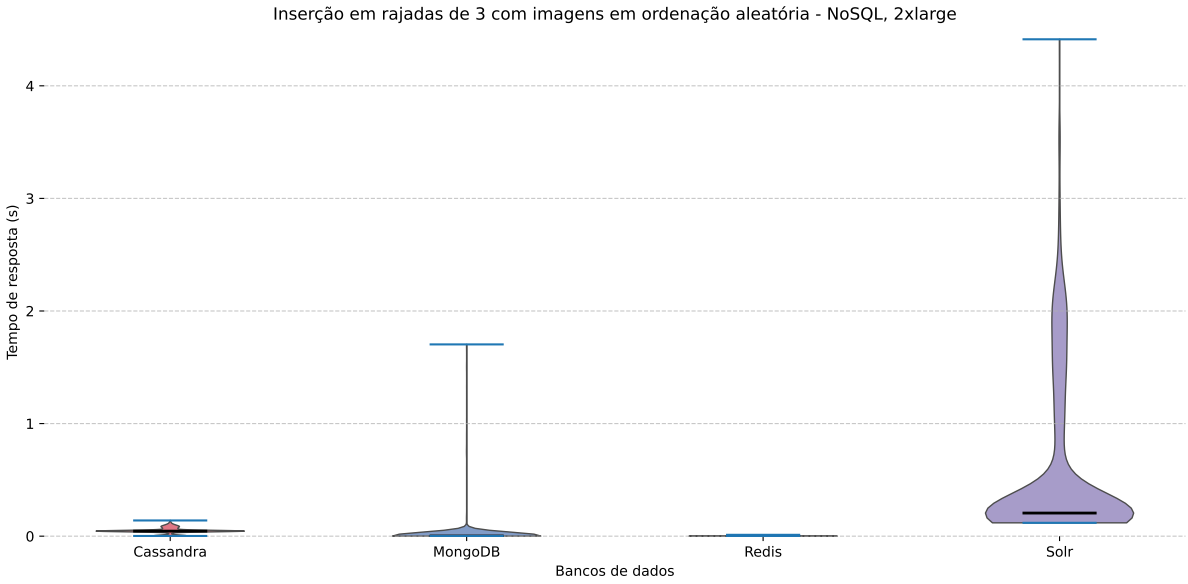

Figura 5.27 – Gráfico dos tempos de resposta (inserção em rajadas, aleatória, NoSQL, 2xlarge)

Fonte: Produzido pelos autores.

Novamente, o Redis teve o melhor resultado em todas as ordenações e máquinas, tanto de tempo de resposta quanto de desvio padrão, enquanto o Solr apresentou os piores.

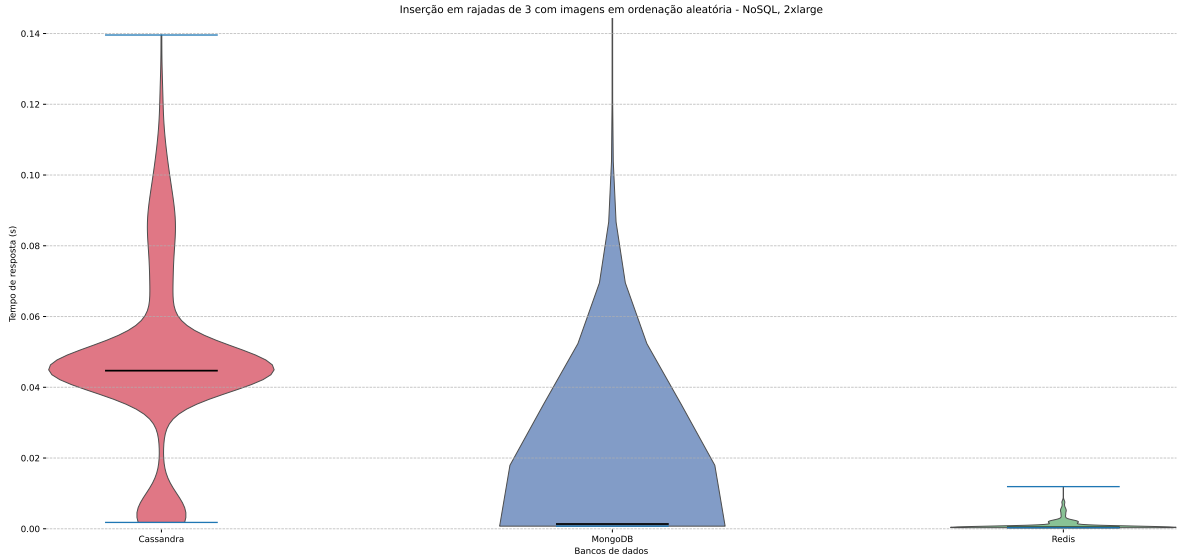

Figura 5.28 – *Zoom* para inspeção das curvas do Cassandra e Redis da Figura 5.27

Fonte: Produzido pelos autores.

Observamos os mesmos altos valores de mudança percentual para o MongoDB na troca de máquinas, embora dessa vez não tenham sido todos de diminuição.

De fato, o único dos bancos de dados que apresentou apenas melhoria dos resultados em relação à mudança de *hardware* foi o Solr (Cassandra também apresentou, mas os dados estão incompletos), possivelmente indicando sua melhoria proporcional ao *hardware* disponível para uso.

Podemos observar, novamente, a tendência do Cassandra à estabilidade, com a maior parte dos valores medidos estando próximos ao centro da curva de distribuição.

A Tabela 5.19 mostra a comparação do tempo de inserção única triplicado com o tempo de inserção em rajadas de 3 imagens. Em azul está destacado o menor tempo dentre os dois comparados.

Para facilitar a visualização, os dados referentes aos testes do Cassandra que não puderam ser realizados em testes de rajada foram omitidos.

Todos os bancos de dados exceto o MongoDB apresentaram melhores resultados ao usar as rajadas em comparação à três inserções únicas. O Cassandra apresentou uma melhoria de quase 50%, indicando uma possível otimização para as operações em rajada, desde que respeitada sua limitação de tamanho por *batch*. Porém, como a carga utilizada nos testes tornou impossível testar duas das três ordenações de imagens, faltam informações para apoiar essa observação.

Tabela 5.19 – Comparação do resultado do teste de inserção única triplicado com o teste de rajadas

Banco de dados e resultados		Ordenação de imagens e máquina utilizada					
		Crescente		Decrescente		Aleatório	
		2xlarge	8xlarge	2xlarge	8xlarge	2xlarge	8xlarge
Cassandra	$t_{m,singular} \times 3$ (s)	-	-	-	-	0.094269	0.084186
	$t_{m,rajada}$ (s)	-	-	-	-	0.047880	0.046000
MongoDB	$t_{m,singular} \times 3$ (s)	0.022683	0.014592	0.015537	0.007929	0.014682	0.014214
	$t_{m,rajada}$ (s)	0.019739	0.011138	0.018498	0.015976	0.017421	0.015521
Redis	$t_{m,singular} \times 3$ (s)	0.001653	0.001548	0.001542	0.001467	0.001470	0.001332
	$t_{m,rajada}$ (s)	0.000896	0.000905	0.000883	0.000758	0.001031	0.000976
Solr	$t_{m,singular} \times 3$ (s)	0.751282	0.748335	0.735246	0.732209	0.765760	0.757966
	$t_{m,rajada}$ (s)	0.573098	0.532842	0.581296	0.531759	0.584428	0.526558

Fonte: Produzido pelos autores.

Na Tabela 5.20, os resultados para o Cassandra foram omitidos, uma vez que apenas uma ordenação foi testada.

Houve grande variação das ordenações que apareceram em cada avaliação, mas podemos observar que todos os casos com melhor tempo médio ocorreram na máquina 8xlarge, que possui *hardware* superior.

Tabela 5.20 – Casos com pior e melhor resultado para cada banco de dados (inserção em rajada, NoSQL)

BD	Pior tempo médio	Melhor tempo médio	Menor distribuição	Maior distribuição
MongoDB	Crescente - 2x	Crescente - 8x	Decrescente - 2x	Aleatório - 2x
Redis	Aleatório - 2x	Decrescente - 8x	Decrescente - 8x	Crescente - 8x
Solr	Aleatório - 2x	Aleatório - 8x	Aleatório - 8x	Decrescente - 2x

Fonte: Produzido pelos autores.

Com o menor e maior valor de tempo de resposta para cada caso, foram elaborados os gráficos das Figuras 5.29 e 5.30.

Para quase todos os bancos de dados na avaliação dos menores tempos de resposta, a máquina 8xlarge teve resultados melhores, que reforça a noção vista anteriormente nos casos de melhor e pior tempo de resposta e distribuição.

Na análise dos maiores tempos de resposta, no entanto, esse paralelo se torna mais difuso, pois enquanto Redis e Solr apresentaram essa mesma redução, Mongo e Cassandra tiveram um aumento.

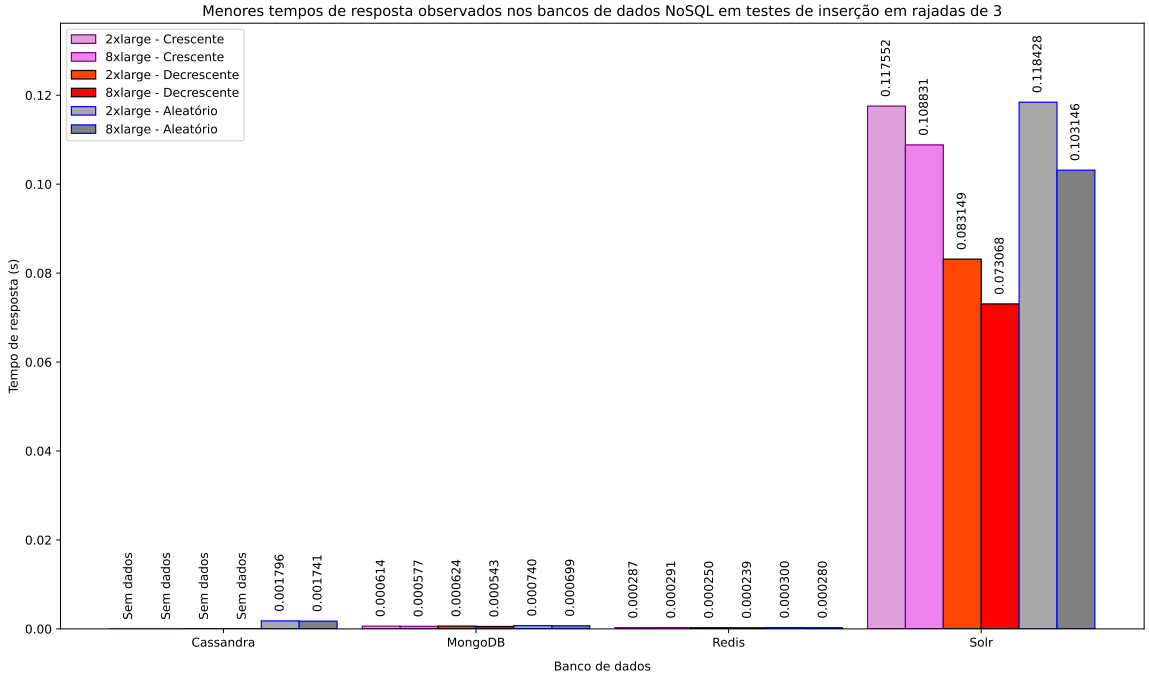

Figura 5.29 – Gráfico do menor tempo de resposta de inserção em rajada em NoSQL

Fonte: Produzido pelos autores.

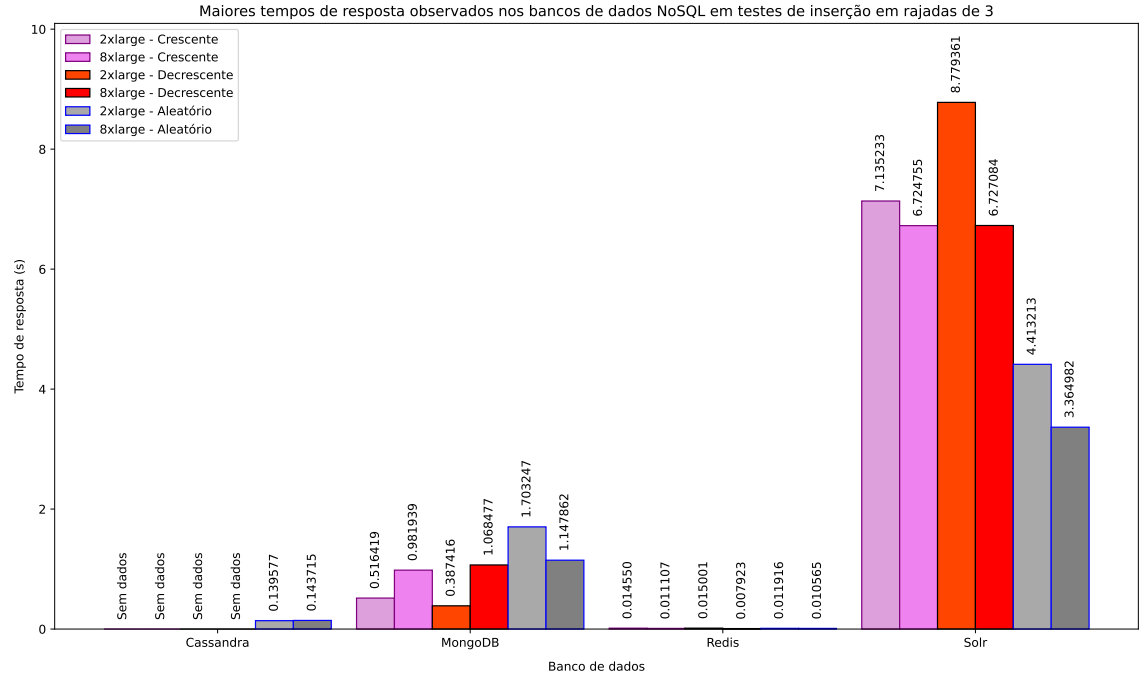

Figura 5.30 – Gráfico do maior tempo de resposta de inserção em rajada em NoSQL

Fonte: Produzido pelos autores.

5.3.3 Testes de deleção

Novamente, os testes de deleção utilizaram a busca por ID para identificar a entrada a ser deletada.

A Tabela 5.21 mostra os resultados obtidos nos testes de deleção, com os melhores resultados por ordenação destacados em azul e os piores em vermelho, e a Tabela 5.22 dispõe a mudança percentual das medidas na troca de máquina.

Tabela 5.21 – Resultados dos testes de deleção dos bancos de dados não relacionais

Banco de dados e resultados		Ordenação de imagens e máquina utilizada					
		Crescente		Decrescente		Aleatório	
		2xlarge	8xlarge	2xlarge	8xlarge	2xlarge	8xlarge
Cassandra	t_m (s)	0.000954	0.000796	0.000739	0.000732	0.000787	0.000693
	σ (s)	0.000438	0.000072	0.000172	0.000072	0.000232	0.000049
MongoDB	t_m (s)	0.000599	0.000640	0.000579	0.000564	0.000676	0.000556
	σ (s)	0.000507	0.000667	0.000145	0.000125	0.000209	0.000130
Redis	t_m (s)	0.000155	0.000145	0.000143	0.000139	0.000139	0.000138
	σ (s)	0.000096	0.000085	0.000086	0.000084	0.000089	0.000083
Solr	t_m (s)	0.022924	0.028313	0.024128	0.023727	0.031077	0.031097
	σ (s)	0.015377	0.023783	0.015080	0.016001	0.013896	0.016289

Fonte: Produzido pelos autores.

Tabela 5.22 – Mudança percentual dos resultados ao trocar da 2xlarge para a 8xlarge (deleção, NoSQL)

Banco de dados e mudança percentual		Ordenação de imagens		
		Crescente	Decrescente	Aleatório
Cassandra	t_m%	-16.56%	-0.95%	-11.94%
	σ%	-83.56%	-58.14%	-78.88%
MongoDB	t_m%	+6.84%	-2.59%	-17.75%
	σ%	+31.56%	-13.79%	-37.80%
Redis	t_m%	-6.45%	-2.80%	-0.72%
	σ%	-11.46%	-2.33%	-6.74%
Solr	t_m%	+23.51%	-1.66%	+0.06%
	σ%	+54.67%	+6.11%	+17.22%

Fonte: Produzido pelos autores.

A distribuição das medidas obtidas para deleção em ordenação crescente na máquina 2xlarge está disposta na Figura 5.31, e sua visão com *zoom* para as curvas do Redis e Cassandra na Figura 5.32.

Os resultados dos outros casos de ordenação e máquinas estão dispostos no Apêndice A, representados pelas Figuras A.86, A.87, A.88, A.89, A.90, A.91, A.92, A.93, A.94 e A.95.

Mais uma vez, o Redis obteve os melhores resultados de tempo médio e distribuição. Em segundo lugar, temos o MongoDB para tempo de resposta, e o Cassandra para a distribuição.

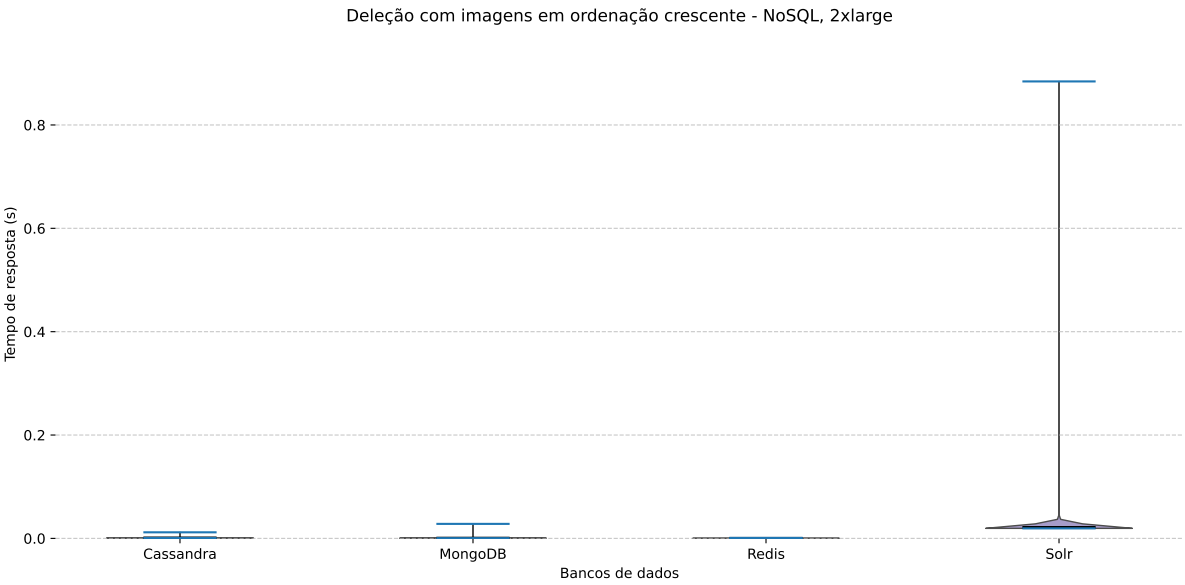

Figura 5.31 – Gráfico dos tempos de resposta (deleção, crescente, NoSQL, 2xlarge)

Fonte: Produzido pelos autores.

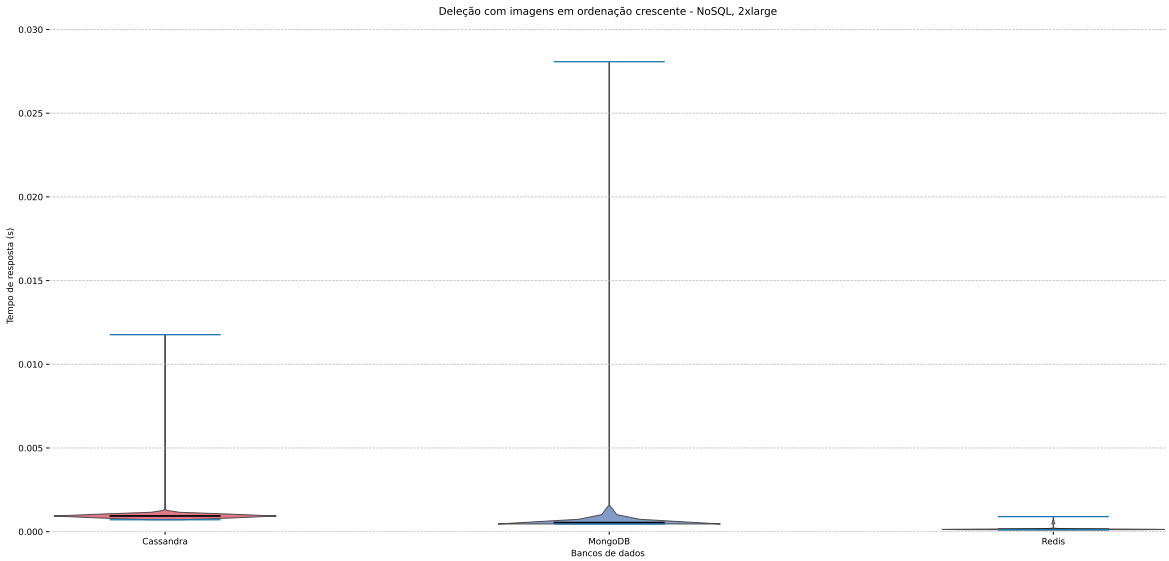

Figura 5.32 – *Zoom* para inspeção das curvas do Cassandra, MongoDB e Redis da Figura 5.31

Fonte: Produzido pelos autores.

O Solr apresentou tempos muito menores se comparados aos resultados de inserção, mas ainda muito distantes dos outros NoSQLs.

É interessante também notar a variação do formato da curva de distribuição de dados do Cassandra para cada caso de teste, fator não observado no MongoDB e no Redis, indicativo da sua forma de estabilização no processamento das *queries*.

Na Tabela 5.23, podemos observar que os testes de ordenação crescente apresentaram consistentemente a maior distribuição de tempos de resposta, fator observado também nos testes de inserção única em NoSQL.

Além disso, o teste aleatório apresentou, na maioria dos casos, não só o melhor tempo médio, como a menor distribuição de dados.

Novamente, podemos observar a predominância da máquina com *hardware* superior nos melhores resultados de tempo médio e distribuição.

Tabela 5.23 – Casos com pior e melhor resultado para cada banco de dados (deleção, NoSQL)

BD	Pior tempo médio	Melhor tempo médio	Menor distribuição	Maior distribuição
Cassandra	Crescente - 2x	Aleatório - 8x	Aleatório - 8x	Crescente - 2x
MongoDB	Aleatório - 2x	Aleatório - 8x	Decrescente - 8x	Crescente - 8x
Redis	Crescente - 2x	Aleatório - 8x	Aleatório - 8x	Crescente - 2x
Solr	Aleatório - 8x	Crescente - 2x	Aleatório - 2x	Crescente - 8x

Fonte: Produzido pelos autores.

Os gráficos das Figuras 5.33 e 5.34 mostram os dados obtidos para menor e maior valor de tempo de resposta de cada caso.

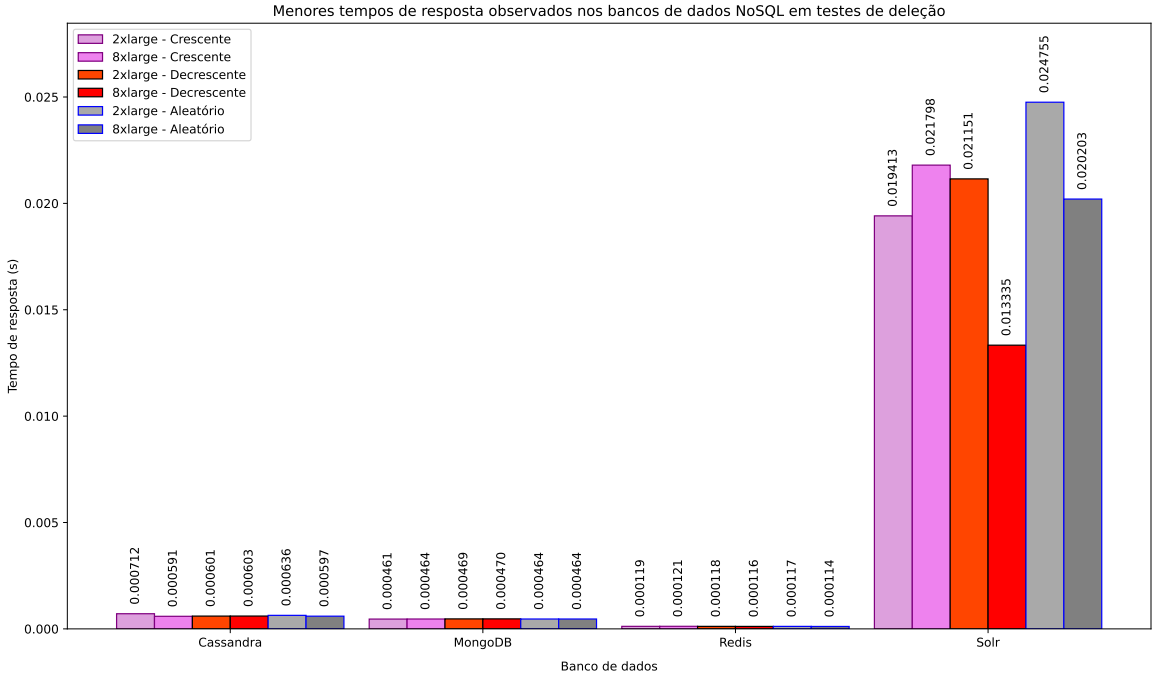

Figura 5.33 – Gráfico do menor tempo de resposta de deleção em NoSQL

Fonte: Produzido pelos autores.

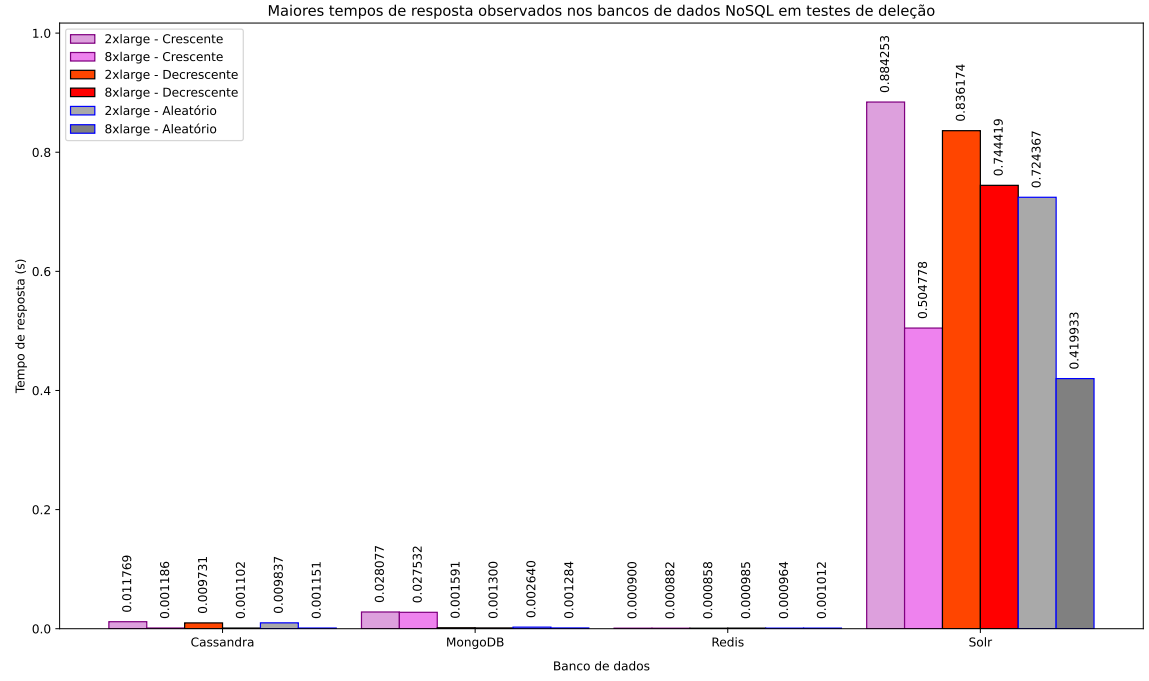

Figura 5.34 – Gráfico do maior tempo de resposta de deleção em NoSQL

Fonte: Produzido pelos autores.

De maneira geral, os bancos de dados NoSQL apresentaram queda dos menores e maiores valores de tempo de resposta ao trocar para a máquina de *hardware* superior, com

atenção especial ao Cassandra e o Solr, que tiveram as maiores quedas.

5.3.4 Testes de leitura

Para os testes de leitura, foi utilizada apenas a ordenação crescente. A Tabela 5.24 mostra os resultados obtidos, bem como a mudança percentual ao trocar de máquina, com o melhor resultado das medidas em azul e os piores em vermelho.

Tabela 5.24 – Resultados dos testes de leitura dos bancos de dados não relacionais

Banco de dados e resultados		Máquina utilizada e mudança percentual		
		2xlarge	8xlarge	%
Cassandra	t_m (s)	0.344312	0.329222	-4.38%
	σ (s)	0.052163	0.049624	-4.87%
MongoDB	t_m (s)	0.001905	0.001783	-6.40%
	σ (s)	0.001332	0.001142	-14.26%
Redis	t_m (s)	0.000401	0.000487	+21.45%
	σ (s)	0.000752	0.001149	+52.79%
Solr	t_m (s)	0.002074	0.001727	-16.73%
	σ (s)	0.000495	0.000183	-63.03%

Fonte: Produzido pelos autores.

A distribuição das medidas pode ser observada nas Figuras 5.35 e 5.37. Devido à escala da curva do Cassandra, as Figuras 5.36 e 5.38 apresentam um *zoom* das curvas do MongoDB, Redis e Solr.

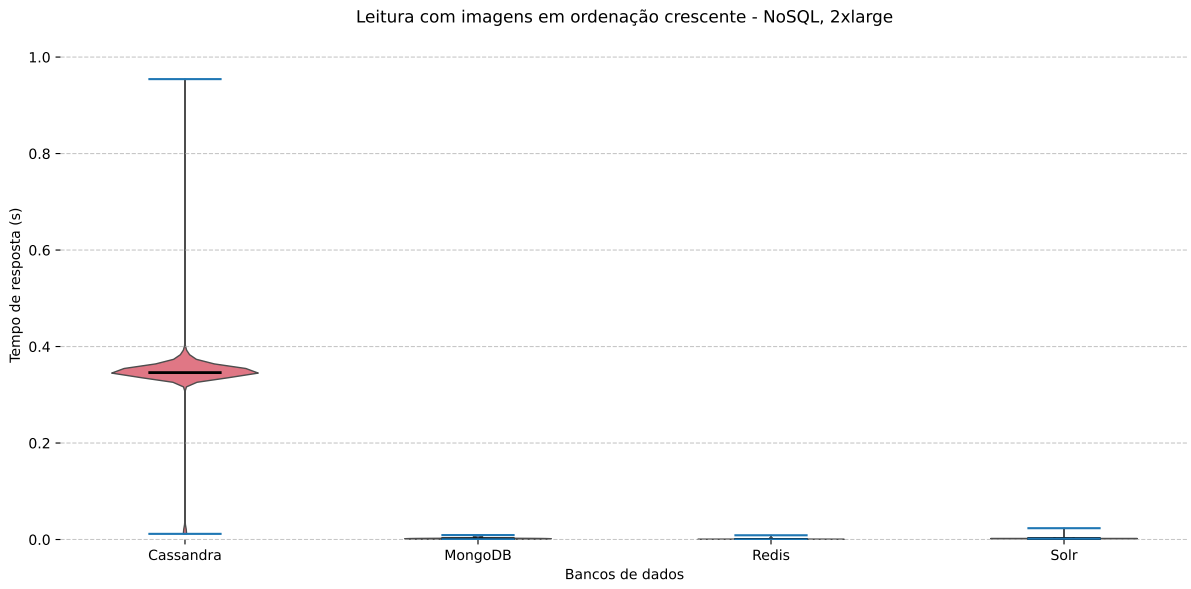

Figura 5.35 – Gráfico dos tempos de resposta (leitura, NoSQL, 2xlarge)

Fonte: Produzido pelos autores.

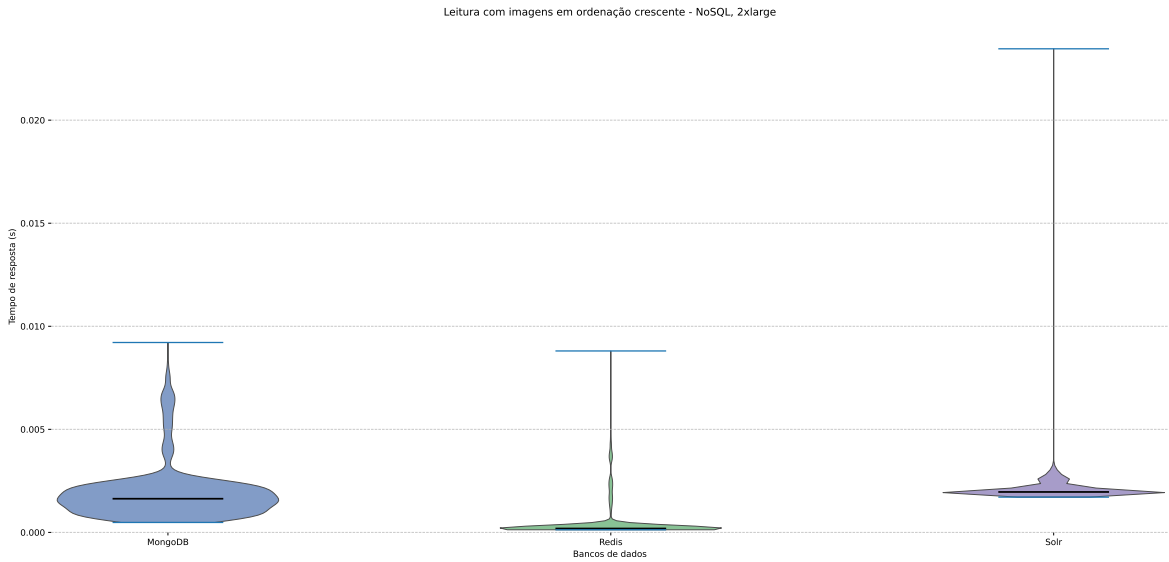

Figura 5.36 – *Zoom* para inspeção das curvas do MongoDB, Redis e Solr da Figura 5.35

Fonte: Produzido pelos autores.

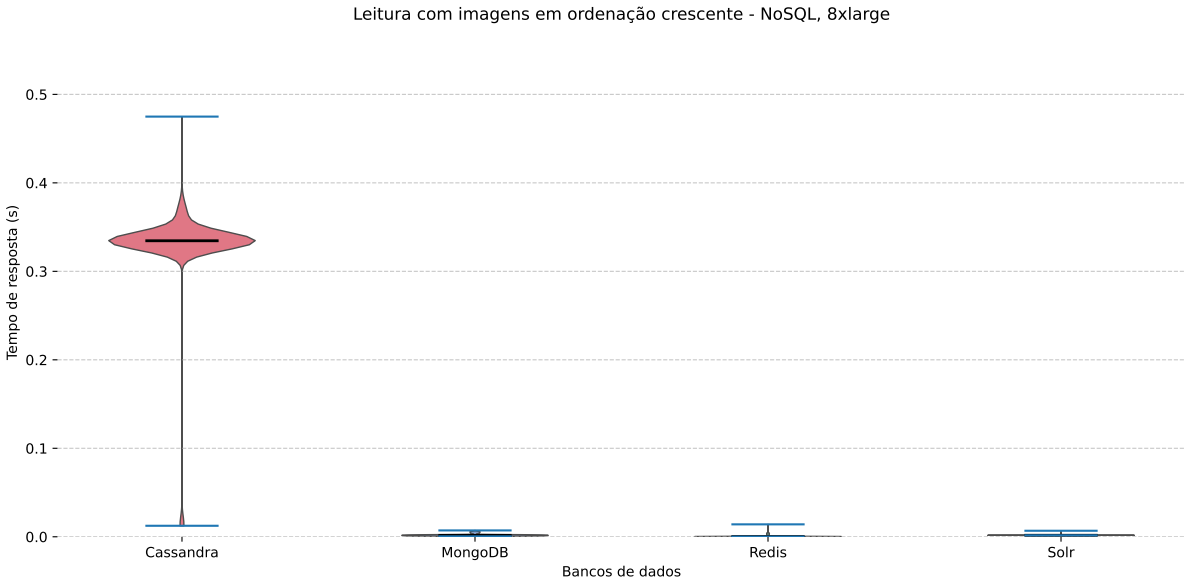

Figura 5.37 – Gráfico dos tempos de resposta (leitura, NoSQL, 8xlarge)

Fonte: Produzido pelos autores.

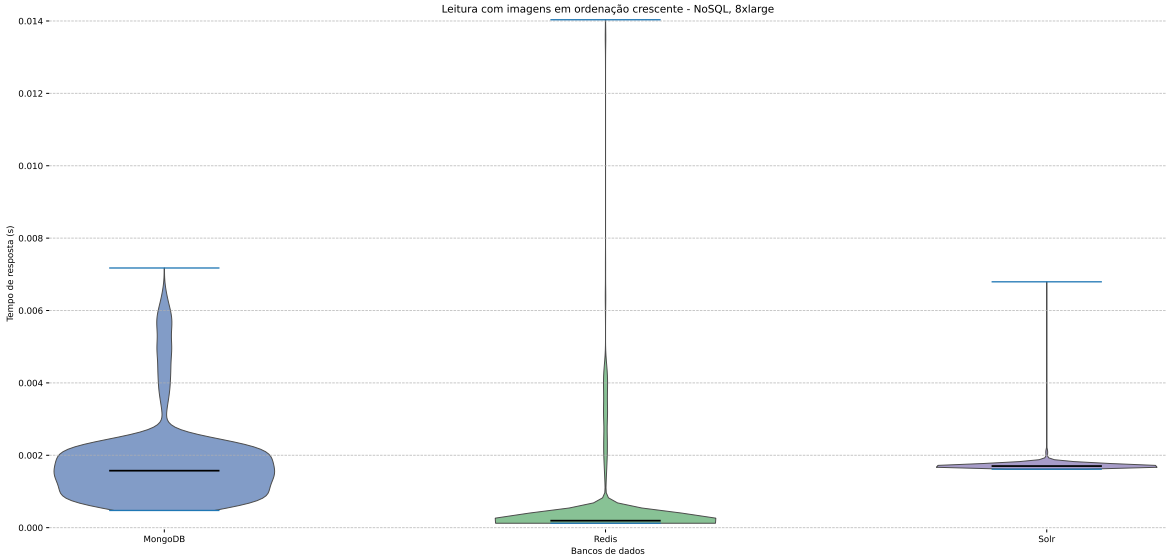

Figura 5.38 – *Zoom* para inspeção das curvas do MongoDB, Redis e Solr da Figura 5.37

Fonte: Produzido pelos autores.

Por fim, as Figuras 5.39 e 5.40 mostram a relação dos menores e maiores valores do tempo de resposta para cada banco de dados.

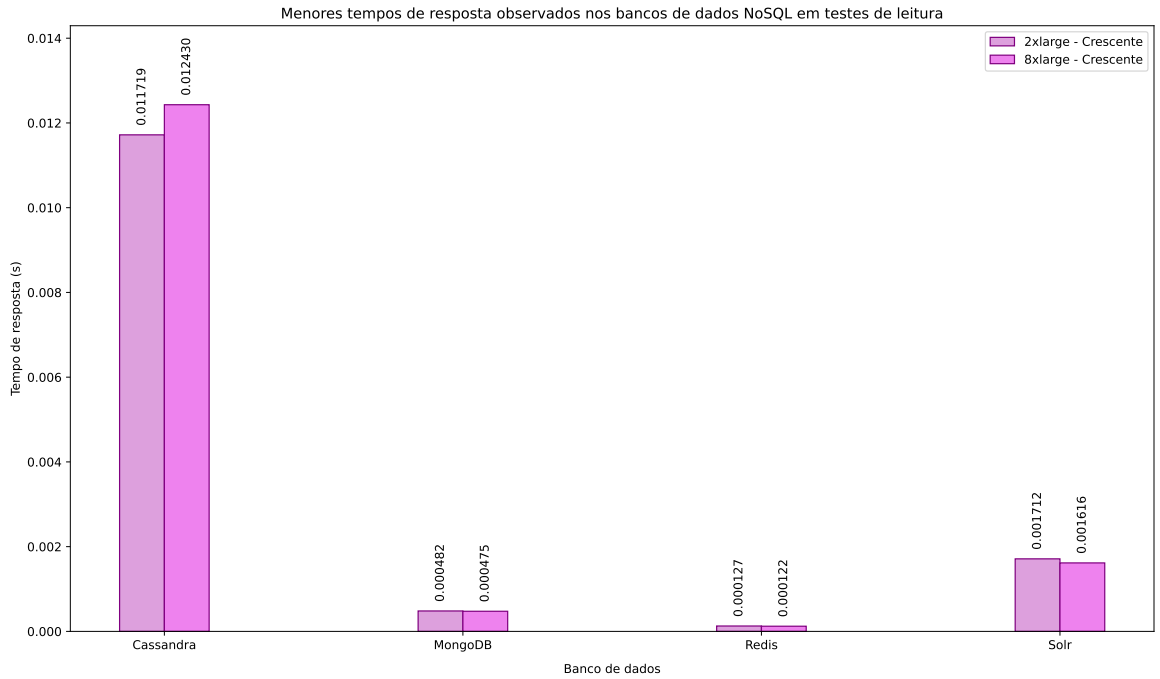

Figura 5.39 – Gráfico do menor tempo de resposta de leitura em NoSQL

Fonte: Produzido pelos autores.

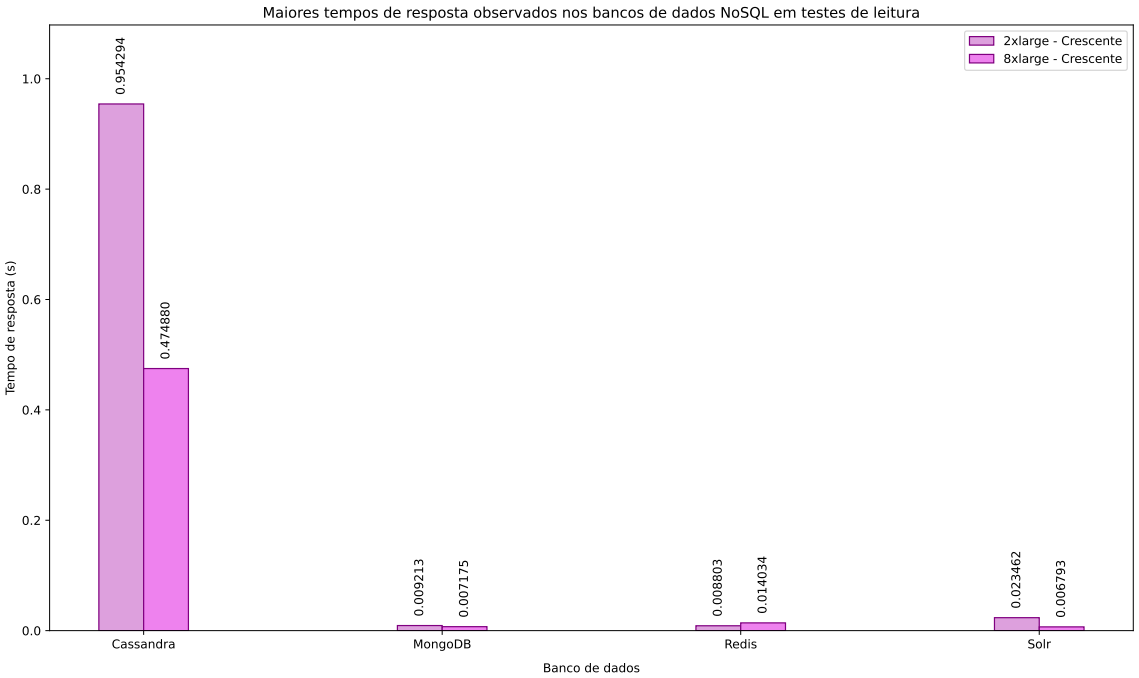

Figura 5.40 – Gráfico do maior tempo de resposta de leitura em NoSQL

Fonte: Produzido pelos autores.

É interessante observar o contraste da resposta do Cassandra à leitura se comparado aos outros casos de teste. Isso se dá pois o Cassandra é um banco de dados otimizado para a escrita, e sua leitura é naturalmente mais lenta.

O Solr, por sua vez, se destacou nos testes de leitura. Embora o Redis continue mantendo o melhor resultado de tempo médio, a distribuição do Solr foi a melhor nos testes de leitura, e seu tempo médio se equiparou ao do MongoDB, até mesmo o superando na máquina 8xlarge. Levando em conta os resultados anteriores, notamos uma certa otimização do Solr para leitura do dados.

Com exceção do Redis, há também uma clara melhoria de performance na troca para a máquina com *hardware* superior, com destaque para a grande melhoria de estabilidade de leitura do Solr.

5.4 Comparação geral

5.4.1 Bancos de dados

A Tabela 5.25 mostra as posições de cada banco de dados para cada caso de teste em função do tempo médio de resposta, enquanto a Tabela 5.26 mostra o mesmo, mas em relação ao desvio padrão. C denota os testes de ordenação crescente, D os de ordenação decrescente e A os de ordenação aleatória. O melhor resultado está destacado em azul, e o

pior está destacado em vermelho.

Tabela 5.25 – Posição dos bancos de dados nos testes por tempo médio de resposta

Banco de dados	Cenário de teste e ordenação testada										
	Inserção única			Inserção em rajada			Deleção			Leitura	
	C	D	A	C	D	A	C	D	A	C	Posição
MariaDB	3	3	3	3	3	3	6	7	6	7	4
Microsoft SQL	4	4	4	4	4	4	4	4	4	3	3
MySQL	7	7	7	6	6	7	8	8	7	2	7
PostgreSQL	5	5	5	5	5	6	5	5	5	6	6
Cassandra	6	6	6	-	-	5	3	3	3	8	5
MongoDB	2	2	2	2	2	2	2	2	2	4	2
Redis	1	1	1	1	1	1	1	1	1	1	1
Solr	8	8	8	7	7	8	7	6	8	5	8

Fonte: Produzido pelos autores.

Tabela 5.26 – Posição dos bancos de dados nos testes por desvio padrão

Banco de dados	Cenário de teste e ordenação testada										
	Inserção única			Inserção em rajada			Deleção			Leitura	
	C	D	A	C	D	A	C	D	A	C	Posição
MariaDB	3	5	4	3	2	4	7	7	7	7	5
Microsoft SQL	4	4	2	4	4	3	4	4	4	5	4
MySQL	7	7	7	6	6	7	8	8	8	4	8
PostgreSQL	5	6	5	5	5	5	6	6	6	6	6
Cassandra	2	3	3	-	-	2	2	2	2	8	2
MongoDB	6	2	6	2	3	6	3	3	3	3	3
Redis	1	1	1	1	1	1	1	1	1	2	1
Solr	8	8	8	7	7	8	5	5	5	1	7

Fonte: Produzido pelos autores.

O Redis foi o banco de dados com maior destaque, ficando em primeiro na comparação tanto de tempo de resposta quanto de distribuição dos dados. Para as posições seguintes, em análise do tempo médio de resposta, temos o MongoDB em segundo e o Microsoft SQL Server em terceiro. Já analisando o desvio padrão, temos o Cassandra em segundo lugar, e o MongoDB em terceiro.

O Redis, por sua característica de banco de dados em memória, liderou os testes durante todo o desenvolvimento da pesquisa, e em uma visão geral, alcançou a posição de melhor banco de dados dentre os testados para o armazenamento de imagens.

O Cassandra, por sua vez, se destacou por sua ótima estabilidade, mesmo não sendo uma das opções mais rápidas. Ao encontrar uma requisição ao *cluster* que precise de um maior tempo de resposta, o conector limita o tempo de todas as requisições seguintes a algo próximo desse limiar, para garantir que todas as informações sejam transmitidas de

forma correta. Isso causa uma grande previsibilidade de cada operação, apesar de diminuir a velocidade média de todas as operações, validando os valores encontrados.

O MongoDB se mostrou muito rápido em geral, e sua estabilidade também ocupou lugar de destaque. Como esperado, o MongoDB é um banco de dados confiável de alto desempenho, pelos testes realizados, se mostra uma excelente opção para o armazenamento de imagens.

O Solr, que originalmente foi cogitado como um banco de dados promissor para o armazenamento de imagens, se demonstrou incompatível para esse fim. Uma vez que diferentes bancos de dados lidam com arquivos binários de diversas maneiras, é possível concluir que o Solr não foi otimizado para lidar com esse tipo de dado.

O Microsoft SQL, mesmo sendo um banco de dados relacional, ocupou posição de destaque pela sua velocidade e estabilidade ao longo de todos os testes. O MariaDB demonstrou resultados bem próximos, perdendo suas posições de destaque especialmente para operações de deleção. Dentre as opções relacionais avaliadas, esses dois podem ser considerados as melhores escolhas.

5.4.2 Comparação entre SQL e NoSQL

As Tabelas 5.27 e 5.28 apresentam, respectivamente, uma comparação percentual entre a média dos resultados de tempo médio e desvio padrão para os dois melhores bancos de dados SQL e NoSQL. IS, IR, D, L, t_m e σ denotam, respectivamente, inserção única, inserção em rajada, deleção, leitura, tempo médio e desvio padrão.

Tabela 5.27 – Diferença percentual do tempo médio entre bancos de dados SQL e NoSQL para cada caso de teste

Posição	Banco de dados	IS (s)	IR (s)	D (s)	L (s)	t_m (s)	Diferença (%)
1º	Microsoft SQL	0.008972	0.032341	0.003042	0.001821	0.011544	-95.68%
	Redis	0.000501	0.000908	0.000143	0.000444	0.000499	
2º	MariaDB	0.006442	0.018267	0.025215	0.016740	0.016666	-64.29%
	MongoDB	0.004980	0.016382	0.000602	0.001844	0.005952	

Fonte: Produzido pelos autores.

Uma vez que bancos de dados costumam ser utilizados em sua totalidade em aplicações reais, fazendo uso de todas as operações disponíveis, as posições de cada BD foram consideradas seguindo as Tabelas 5.25 e 5.26, e os tempos foram calculados por média geral, sem atribuição de pesos para cada tipo de operação.

Tabela 5.28 – Diferença percentual do desvio padrão entre bancos de dados SQL e NoSQL para cada caso de teste

Posição	Banco de dados	IS (s)	IR (s)	D (s)	L (s)	σ (s)	Diferença (%)
1º	Microsoft SQL	0.027799	0.071317	0.001486	0.004000	0.026151	-96.52%
	Redis	0.000973	0.001636	0.000087	0.000950	0.000911	
2º	MariaDB	0.027014	0.065749	0.065259	0.044601	0.050656	-52.06%
	Cassandra	0.021456	0.024606*	0.000173	0.050894	0.024282	

Fonte: Produzido pelos autores.

*Não foi possível testar o Cassandra com todas as ordenações para o caso de inserção em rajadas, então esse dado pode estar um pouco distoante.

A diferença de performance dos dois melhores bancos de dados NoSQL em relação aos dos melhores SQL é grande, e deixa bem em evidência as vantagens de utilização de bancos de dados não relacionais para sistemas que cuidam do armazenamento de imagens nos próprios BDs.

Embora o Redis tenha apresentado resultados muito além de todos os outros testados, com sua melhoria percentual estando acima de 95%, mesmo ao comparar os BDs relacionais com o Cassandra e o MongoDB, que tiveram ótimos resultados, mas não tão distantes quanto o Redis, temos melhoria no tempo de resposta e na distribuição acima de 50%, um resultado que ilustra bem as vantagens do uso dos NoSQL.

5.4.3 Comparação entre ordenações

A partir dos dados coletados para qual ordenação apresentou os piores e melhores tempos médios e distribuições nos testes, foi elaborado um gráfico de setores para analisar a proporção de aparição de cada uma das ordenações nessas quatro categorias, e esse está disposto na Figura 5.41. Foram considerados tanto os testes realizados nos BDs relacionais quanto nos não relacionais.

As Figuras 5.42 e 5.43 apresentam uma visão semelhante, mas se restringindo aos testes SQL e NoSQL, respectivamente.

Os resultados específicos para cada um dos testes realizados (inserção única, inserção em rajada e deleção) estão dispostos no Apêndice B, representados pelas Figuras B.96, B.97 e B.98.

Analisando todos os gráficos, é possível notar uma certa tendência da ordenação aleatória em apresentar a melhor distribuição, mas também o pior tempo médio. Ao mesmo tempo, podemos também observar que a ordenação crescente costuma apresentar a pior distribuição.

O melhor tempo médio não exibe um padrão claro, tendo muitas variações entre os casos, de forma que não podemos tirar uma conclusão sólida a respeito das ordenações.

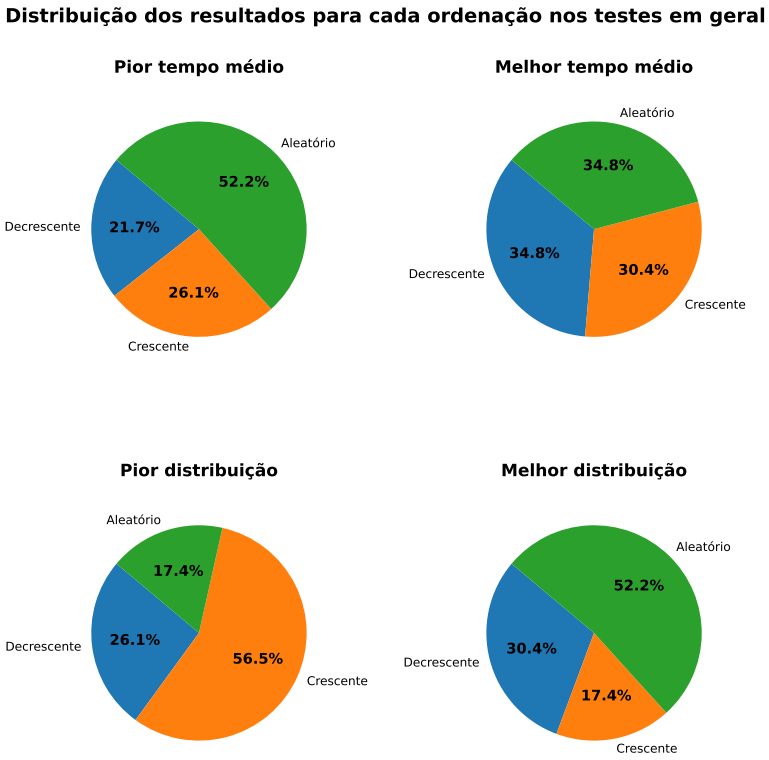

Figura 5.41 – Gráfico da distribuição das ordenações por casos analisados em geral

Fonte: Produzido pelos autores.

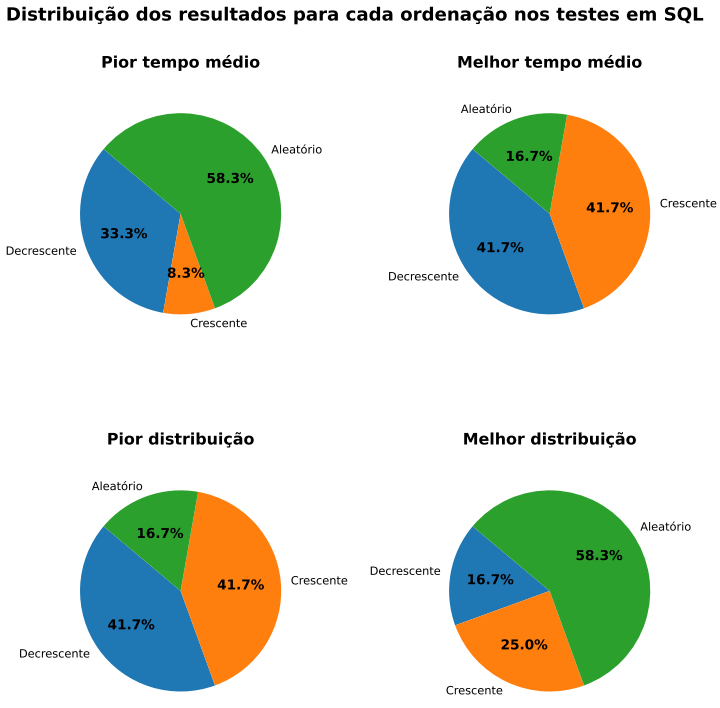

Figura 5.42 – Gráfico da distribuição das ordenações por casos analisados em SQL

Fonte: Produzido pelos autores.

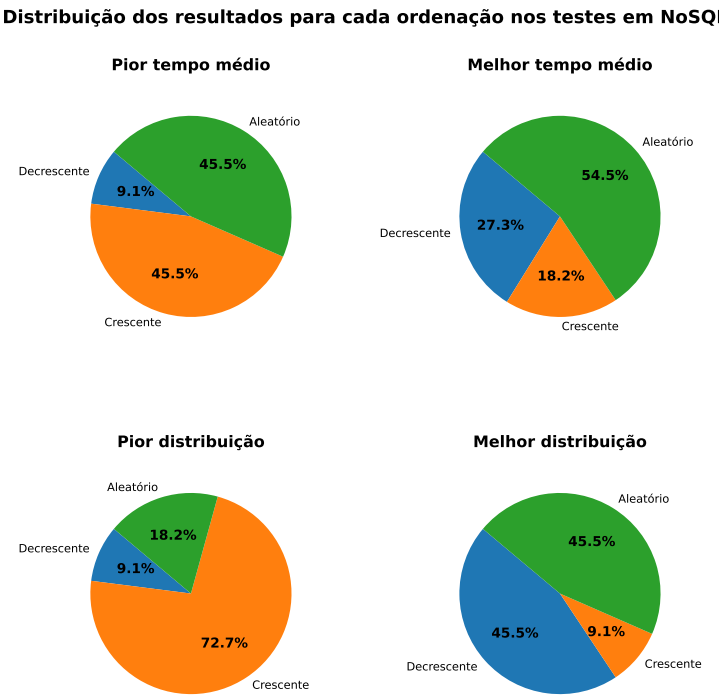

Figura 5.43 – Gráfico da distribuição das ordenações por casos analisados em NoSQL

Fonte: Produzido pelos autores.

Para os SQLs, observamos que a ordenação aleatória apresentou maior proporção no pior tempo médio e na melhor distribuição, enquanto as ordenações crescente e decrescente se destacaram no melhor tempo médio com a pior distribuição. Porém, a ordenação crescente apresentou o pior tempo médio em apenas 8.3% das vezes, enquanto a decrescente apresentou 33%.

Para os NoSQLs, podemos perceber que a ordenação crescente apresentou os piores resultados, com 45.5% de pior tempo médio e 72.7% de pior distribuição. Já a aleatória, embora também com 45.5% de pior tempo médio, atingiu 54.5% de melhor tempo e 45.5% melhor distribuição.

A grande proporção de casos de pior tempo médio para a ordenação aleatória destaca os resultados da ordenação decrescente, que embora com menor número de casos de melhor tempo médio, apresentou a mesma proporção que a aleatória para melhor distribuição, e teve bem menos participação nos piores tempos, com apenas 9.1%.

A inserção única apresentou resultados conflitantes, e uma conclusão satisfatória se mostrou difícil de extrair. Para a inserção em rajadas, no entanto, podemos ver que a ordenação aleatória apresentou os piores resultados, e com a grande proporção da crescente nas piores distribuições, é interessante considerar uma ordenação decrescente para esse tipo de operação.

A deleção apresenta a ordenação decrescente com as menores proporções de melhores

resultados, mas com participação levemente maior nos piores resultados. Para atingir melhor performance nas operações de deleção, se mostra necessário escolher qual fator dentre tempo médio e distribuição desses seria mais desejável.

Com essas informações, é possível observar que, para maior balanço entre tempo médio e distribuição das medidas, a ordenação decrescente se mostra interessante, embora não por uma margem muito grande.

5.4.4 Comparação entre máquinas

As Figuras 5.44 e 5.45 apresentam a proporção da melhoria dos resultados de tempo médio e desvio padrão ao trocar da máquina 2xlarge para a 8xlarge.

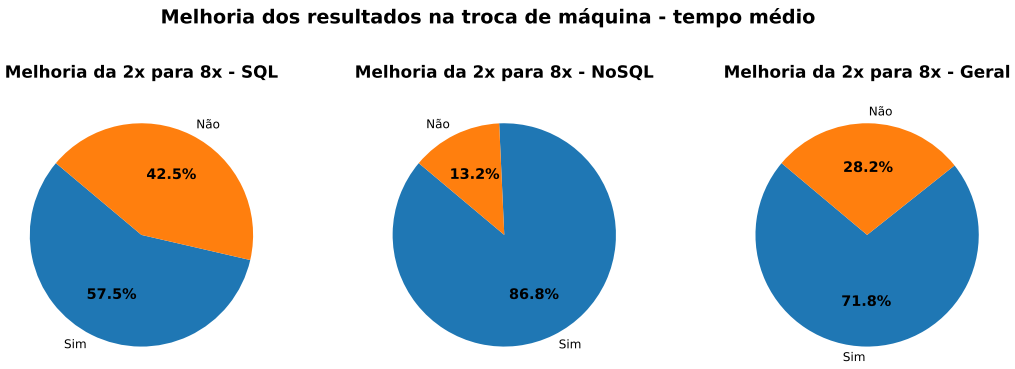

Figura 5.44 – Gráfico da melhoria de tempo médio na troca de máquina

Fonte: Produzido pelos autores.

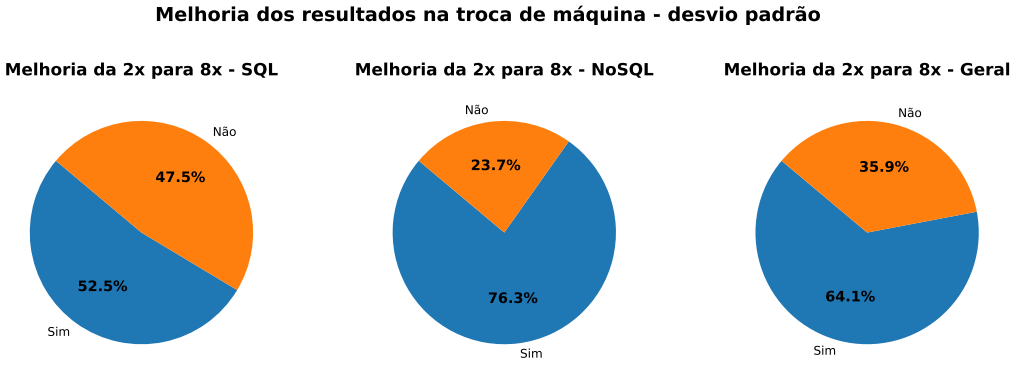

Figura 5.45 – Gráfico da melhoria de desvio padrão na troca de máquina

Fonte: Produzido pelos autores.

Observamos uma melhora geral nos resultados, com importante destaque para o fato de que essa melhoria foi bem mais acentuada nos bancos de dados não relacionais, enquanto os SQLs apresentaram proporções quase iguais para os dois casos. Assim, podemos concluir

que a troca de *hardware* teve um impacto positivo bem mais significativo nos resultados dos bancos NoSQL.

Em vista do que foi apresentado, podemos perceber as grandes vantagens do uso de bancos de dados não relacionais para o armazenamento de imagens. Em SQL algumas escolhas são otimizadas para binários, mas a comparação de otimização NoSQL deixa clara a motivação de sua utilização para lidar com esse tipo de arquivos. As melhores opções NoSQL se demonstraram consideravelmente superiores às melhores opções SQL.

6 Conclusões

Este capítulo condensa todas as informações expostas ao longo do trabalho, relacionando as propostas iniciais com os resultados obtidos. Além disso, discorremos a respeito do que futuros trabalhos poderiam fazer para expandir a pesquisa aqui descrita e chegar a novos resultados.

Tendo a proposta incial do trabalho observar e metrificar o desempenho de bancos de dados quando utilizados para o armazenamento de imagens (e arquivos binários no geral), tivemos o desafio de obter o mais fielmente possível resultados que refletem a utilização do banco de dados em si, diminuindo váriaveis externas. Nossa metodologia incluiu testes extensivos em cima de cada banco de dados, rodados em *scripts* de autoria própria (disponíveis em Miranda e Albuquerque (2023)) com o mínimo de interferência, em *containers* virtualizados. Feitos testes de inserção, inserção em sequência, leitura e deleção, obtendo resultados específicos tanto em SQL quanto NoSQL para a utilização de binários. O melhor resultado obtido, em tempo médio e desvio padrão, foi com a utilização do Redis, que possui protocolos focados em disponibilidade, apesar de ter alguns problemas na consistência do armazenamento, quando lida com múltiplas instâncias. O MongoDB ainda é uma ótima opção que garante melhor segurança na consistência de dados, ainda ficando em vantagem na velocidade média de resposta.

Com um grande objetivo de mensurar objetivamente o conhecimento existente para a utilização de bancos de dados, dito dos modelos NoSQL sendo melhores que os SQLs para armazenamento de imagem, pudemos observar que não é sempre o caso. Com o Solr, um banco de de dados NoSQL orientado a documentos, tivemos tempos de resposta muito inferiores mesmo quando comparado aos SQLs, devido ao tipo de armazenamento e utilização desse modelo, sendo otimizado para busca e manutenção de volumes textuais. O MariaDB e o Microsoft SQL apresentaram desempenhos bons com o gerenciamento de binários, superando algumas das opções NoSQLs nos casos de teste.

Considerando um modelo NoSQL otimizado para o armazenamento de arquivos binários, como o Redis, temos uma eficiência duas ordens de magnitude maiores quando comparado ao modelo básico SQL, e uma ordem de magnitude maior mesmo para SQLs otimizados. Entre o Redis e o Microsoft SQL, as inserções ocorrem 16,93 vezes mais rápidas e as leituras 3,98 vezes mais rápidas. Pode ser observado que as duas primeiras posições ficaram com modelos NoSQL, tendo ainda mais um dos NoSQLs aparecendo em terceiro para desvio padrão.

Temos resultados satisfatórios para mensurar o quão mais eficientes bancos de dados NoSQL se comportam quando comparados aos SQLs, tendo também escolhas NoSQL que

não se adequam ao caso de uso estudado, por consequência de seu próprio foco de arquitetura. Em uma escolha para um projeto que precise armazenar, utilizar e atualizar imagens de forma constante e volumosa, ainda deve-se olhar primeiro às soluções NoSQL que possuam conectores ao sistema projetado, com a melhor opção observada nesse estudo sendo o Redis.

6.1 Trabalhos Futuros

Os resultados encontrados ao final, embora tenham sido satisfatórios, mostrando até mesmo alguns dados surpreendentes, podem ser ainda mais desenvolvidos. Existem diversos bancos de dados, relacionais e não relacionais, que podem ser testados e comparados, fora outros cenários de teste que podem ser explorados. Algumas possibilidades incluem:

- **Extensão dos testes à mais bancos de dados**: com a inclusão de mais bancos de dados, como Oracle, Db2 e Azure, como exemplos de bancos relacionais, e Amazon S3, OrientDB, ArcadeDB, como exemplos de bancos não relacionais, é possível que haja uma completa reordenação dos melhores BDs estipulados nesse trabalho;
- **Novos casos de teste**: ao incluir novos casos de teste, como atualizar uma entrada do banco de dados para substituir uma imagem por outra, é possível obter resultados ainda mais precisos para os bancos aqui estudados, podendo destacar suas vantagens em outros tipos de operação;
- **Inclusão de variação de carga do ambiente**: em nosso trabalho, utilizamos máquinas AWS dedicadas exclusivamente aos testes, mas na prática, bancos de dados são operados em servidores com movimento constante; assim, a variação da carga do ambiente durante as operações poderia revelar dados a respeito de quais bancos de dados são mais resilientes, e portanto, mais adequados para determinados perfis de carga. Também seria interessante incluir cenários com um volume de imagens grande o suficiente para colocar a máquina em sobrecarga, assim estudando o comportamento de cada banco de dados ao trabalhar em um cenário com falta de recursos;
- **Utilização de *clusters* para verificação de escalabilidade**: ao aplicar uma estrutura mais complexa, que faz uso de *clustering*, seria possível avaliar a escalabilidade dos bancos de dados analisados, fator esse muito interessante pois sistemas reais tendem a crescer continuamente, o que faz com que um banco de dados altamente escalável seja preferível ao pensar em soluções para o mercado.

Referências

AMAZON AWS. **Amazon Aurora**. Disponível em: <https://aws.amazon.com/rds/aurora/>. Acesso em: 24 jul. 2023. Citado na p. 30.

AMAZON AWS. **Amazon S3**. Disponível em: <https://aws.amazon.com/s3/>. Acesso em: 5 fev. 2023. Citado na p. 30.

ARAUJO, L. C. **Como customizar o abnTeX2**. 2015. Wiki do abnTeX2. Disponível em: <https://github.com/abntex/abntex2/wiki/ComoCustomizar>. Acesso em: 27 abr. 2015. Citado na p. 21.

ARCADEDB. **ArcadeDB**. Disponível em: <https://arcadedb.com/>. Acesso em: 24 jul. 2023. Citado na p. 30.

AWS. **What is a Columnar Database?** Disponível em: <https://aws.amazon.com/nosql/columnar/>. Acesso em: 7 mai. 2023. Citado na p. 32.

AWS. **What Is an In-Memory Database?** Disponível em: <https://aws.amazon.com/nosql/in-memory/>. Acesso em: 7 mai. 2023. Citado na p. 33.

AWS, A. **Tipos de instância**. Disponível em: <https://aws.amazon.com/pt/ec2/instance-types/>. Acesso em: 2 jul. 2023. Citado na p. 38.

BASTIÃO SILVA, L.; BEROUD, L.; COSTA, C.; OLIVEIRA, J. Medical imaging archiving: A comparison between several NoSQL solutions. In: p. 65–68. ISBN 978-1-4799-2131-7. DOI: 10.1109/BHI.2014.6864305. Citado na p. 27.

BATHLA, G.; RANI, R.; AGGARWAL, H. Comparative study of NoSQL databases for big data storage. **International Journal of Engineering Technology**, v. 7, mar. 2018. DOI: 10.14419/ijet.v7i2.6.10072. Citado na p. 26.

BOICEA, A.; RADULESCU, F.; AGAPIN, L. I. MongoDB vs Oracle – Database Comparison. In: 2012 Third International Conference on Emerging Intelligent Data and Web Technologies. 2012. P. 330–335. DOI: 10.1109/EIDWT.2012.32. Citado na p. 27.

CASSANDRA. **Apache Cassandra**. Disponível em: <https://cassandra.apache.org/_/index.html>. Acesso em: 5 fev. 2023. Citado na p. 30.

CHOPADE, M. R. M.; DHAVASE, N. S. Mongodb, couchbase: Performance comparison for image dataset. In: 2017 2nd International Conference for Convergence in Technology (I2CT). 2017. P. 255–258. DOI: 10.1109/I2CT.2017.8226131. Citado na p. 27.

CHRIS SAXON. **How to Delete Millions of Rows Fast with SQL**. Disponível em: <https://blogs.oracle.com/sql/post/how-to-delete-millions-of-rows-fast-with-sql>. Acesso em: 10 jul. 2023. Citado na p. 47.

COOPER, B. F.; SILBERSTEIN, A.; TAM, E.; RAMAKRISHNAN, R.; SEARS, R. Benchmarking Cloud Serving Systems with YCSB. In: PROCEEDINGS of the 1st ACM Symposium on Cloud Computing. Indianapolis, Indiana, USA: Association for Computing Machinery, 2010. (SoCC '10), p. 143–154. ISBN 9781450300360. DOI: 10.1145/1807128.1807152. Disponível em: <https://doi.org/10.1145/1807128.1807152>. Citado na p. 28.

COOPER, B. F. **Yahoo! Cloud Serving Benchmark - Github**. Disponível em: <https://github.com/brianfrankcooper/YCSB>. Acesso em: 5 fev. 2023. Citado na p. 35.

DATASTAX. **cassandra.yaml configuration file**. Disponível em: <https://docs.datastax.com/eol/en/dse/6.0/dse-dev/datastax_enterprise/config/configCassandra_yaml.html>. Acesso em: 9 jul. 2023. Citado na p. 59.

DATASTAX. **DataStax Python Driver for Apache Cassandra - Prepared Statements**. Disponível em: <https://docs.datastax.com/en/developer/python-driver/3.21/getting_started/>. Acesso em: 7 mai. 2023. Citado na p. 32.

DATAVERSITY. **NoSQL Data Architecture Data Governance: Everything You Need to Know**. Acesso em 28 fev. 2023. 2016. Disponível em: <https://www.dataversity.net/nosql-data-architecture-data-governance-everything-need-know/>. Acesso em: 28 mar. 2023. Citado na p. 24.

DOCKER HUB. **Docker Hub**. 2023. Disponível em: <https://hub.docker.com/>. Acesso em: 4 fev. 2023. Citado na p. 29.

ELMASRI, R.; NAVATHE, S. Fundamentals of Database Systems. In: 7. ed.: Pearson, 2016. Citado nas pp. 18, 21, 24, 25, 34.

GYŐRÖDI, C.; GYŐRÖDI, R.; PECHERLE, G.; OLAH, A. A comparative study: MongoDB vs. MySQL. In: 2015 13th International Conference on Engineering of Modern Electric Systems (EMES). 2015. P. 1–6. DOI: 10.1109/EMES.2015.7158433. Citado na p. 28.

HAJJAJI, Y.; FARAH, I. R. Performance investigation of selected NoSQL databases for massive remote sensing image data storage. In: 2018 4th International Conference on Advanced Technologies for Signal and Image Processing (ATSIP). 2018. P. 1–6. DOI: 10.1109/ATSIP.2018.8364508. Citado na p. 26.

HARRINGTON, J. L. **Relational database design and implementation clearly explained**. 3rd ed. Amsterdam ; Morgan Kaufmann/Elsevier, 2009. ISBN 9780123747303. Citado nas pp. 21, 22.

HARRISON, G. **Next Generation Databases: NoSQL, NewSQL, and Big Data: What every professional needs to know about the future of databases in a world of NoSQL and Big Data**. Apress, 2016. ISBN 978-1-484-21330-8. Citado na p. 18.

HEIN, N.; BLANKENBACH, J. Evaluation of a NoSQL Database for Storing Big Geospatial Raster Data. **GI$_F$*orum***, v. 1, p. 76–84, jan. 2021. DOI: 10.1553/giscience2021_01_s76. Citado na p. 26.

IBM. **DB2**. Disponível em: <https://www.ibm.com/products/db2/>. Acesso em: 24 jul. 2023. Citado na p. 30.

IEEE XPLORE. **IEEE Xplore**. 2023. Disponível em: <https://ieeexplore.ieee.org/Xplore/home.jsp>. Acesso em: 4 fev. 2023. Citado na p. 29.

ILSVRC2015. **ImageNet Large Scale Visual Recognition Challenge 2015 (ILSVRC2015)**. Disponível em: <https://www.image-net.org/challenges/LSVRC/2015/index.php>. Acesso em: 6 fev. 2023. Citado na p. 35.

KEN W. ALGER. **Choosing a good Shard Key in MongoDB**. Acesso em 28 fev. 2023. 2017. Disponível em: <https://www.kenwalger.com/blog/nosql/mongodb/choosing-good-shard-key-mongodb/>. Acesso em: 28 mar. 2023. Citado na p. 22.

LI, Y.; MANOHARAN, S. A performance comparison of SQL and NoSQL databases. In: p. 15–19. DOI: 10.1109/PACRIM.2013.6625441. Citado na p. 28.

LIU, Y.; CHEN, B.; HE, W.; FANG, Y. Massive image data management using HBase and MapReduce. In: 2013 21st International Conference on Geoinformatics. 2013. P. 1–5. DOI: 10.1109/Geoinformatics.2013.6626187. Citado na p. 26.

MANUAL DO MONGODB. **Query Optimization - MongoDB Documentation**. Acesso em 28 jan. 2023. 2023. Disponível em: <https://www.mongodb.com/docs/manual/core/query-optimization/>. Acesso em: 28 jan. 2023. Citado na p. 27.

MARIADB. **MariaDB**. Disponível em: <https://mariadb.org/>. Acesso em: 5 fev. 2023. Citado na p. 30.

MARIADB. **MariaDB vs MySQL**. Disponível em: <https://mariadb.com/database-topics/mariadb-vs-mysql/>. Acesso em: 11 mai. 2023. Citado na p. 31.

MARTINS, P.; ABBASI, M.; SÁ, F. A Study over NoSQL Performance. In: ROCHA, Á.; ADELI, H.; REIS, L. P.; COSTANZO, S. (Ed.). **New Knowledge in Information Systems and Technologies**. Cham: Springer International Publishing, 2019. P. 603–611. ISBN 978-3-030-16181-1. Citado na p. 27.

MICROSOFT. **Introducing SQL Server Big Data Clusters**. Disponível em: <https://learn.microsoft.com/en-us/sql/big-data-cluster/big-data-cluster-overview?view=sql-server-ver15>. Acesso em: 11 mai. 2023. Citado na p. 31.

MICROSOFT. **SQL Server 2019**. Disponível em: <https://www.microsoft.com/en-us/sql-server/sql-server-2019>. Acesso em: 11 mai. 2023. Citado na p. 31.

MICROSOFT SQL SERVER. **Microsoft SQL Server**. Disponível em: <https://www.microsoft.com/en-us/sql-server>. Acesso em: 5 fev. 2023. Citado na p. 30.

MIRANDA, L.; ALBUQUERQUE, C. **SQL and NoSQL Database Benchmarking for Image Payload**. Disponível em: <https://github.com/leevanf/tcc-database-image-benchmarking>. Acesso em: 12 dez. 2023. Citado nas pp. 35, 36, 79.

MONGODB. **GridFS**. Disponível em: <https://www.mongodb.com/docs/manual/core/gridfs/>. Acesso em: 7 mai. 2023. Citado na p. 33.

MONGODB. **MongoDB**. Disponível em: <https://www.mongodb.com/>. Acesso em: 5 fev. 2023. Citado na p. 30.

MONGODB. **Why Use MongoDB and When to Use It?** Disponível em: <https://www.mongodb.com/why-use-mongodb>. Acesso em: 5 fev. 2023. Citado na p. 33.

MOZILLA. **Blob**. Acesso em 12 fev. 2023. 2023. Disponível em: <https://developer.mozilla.org/en-US/docs/Web/API/Blob>. Acesso em: 12 fev. 2023. Citado na p. 18.

MUKHERJEE, S. The battle between NoSQL Databases and RDBMS, mai. 2019. DOI: 10.15680/IJIRSET.2019.0805107. Citado na p. 28.

MYSQL. **MySQL**. Disponível em: <https://www.mysql.com/>. Acesso em: 5 fev. 2023. Citado na p. 30.

NAGAPPA, S.; DIVYA, S. Bigdata: A Survey On RDBMS And Various NOSQL Databases On Storing Medical Images, ago. 2017. Citado na p. 27.

OLIVEIRA ASSIS, J. de; SOUZA, V. C. O.; PAULA, M. M. V.; CUNHA, J. B. S. Performance evaluation of NoSQL data store for digital media. In: 2017 12th Iberian Conference on Information Systems and Technologies (CISTI). 2017. P. 1–6. DOI: 10.23919/CISTI.2017.7975844. Citado na p. 28.

ORACLE. **Oracle Database**. Disponível em: <https://www.oracle.com/database/>. Acesso em: 24 jul. 2023. Citado na p. 30.

ORACLE CLOUD. **What is MySQL?** Disponível em: <https://www.oracle.com/mysql/what-is-mysql/>. Acesso em: 11 mai. 2023. Citado na p. 31.

PETE SCOTT. **Is MongoDB Open Source? Is Planet Earth Flat?** Abr. 2023. Disponível em: <https://www.percona.com/blog/is-mongodb-open-source>. Acesso em: 11 mai. 2023. Citado na p. 33.

POSTGRESQL. **About PostgreSQL**. Disponível em: <https://www.postgresql.org/about/>. Acesso em: 11 mai. 2023. Citado na p. 32.

POSTGRESQL. **PostgreSQL**. Disponível em: <https://www.postgresql.org/>. Acesso em: 5 fev. 2023. Citado na p. 30.

REBECCA, R. D.; SHANTHI, E. I. A NoSQL Solution to efficient storage and retrieval of Medical Images. **International Journal of Scientific amp; Engineering Research**, v. 7, n. 2, p. 545–549, fev. 2016a. Citado na p. 27.

REBECCA, R. D.; SHANTHI, E. I. Analysing the suitability of storing Medical Images in NoSQL Databases. **International Journal of Scientific amp; Engineering Research**, v. 7, n. 8, p. 527–531, ago. 2016b. Citado na p. 27.

REDIS. **Introduction to Redis**. Disponível em: <https://redis.io/docs/about/>. Acesso em: 7 mai. 2023. Citado na p. 33.

REDIS. **Redis**. Disponível em: <https://redis.io/>. Acesso em: 5 fev. 2023. Citado na p. 30.

RESEARCHGATE. **ResearchGate**. 2023. Disponível em: <https://www.researchgate.net/>. Acesso em: 4 fev. 2023. Citado na p. 29.

SADALAGE, P. J.; FOWLER, M. **NoSQL Distilled: A Brief Guide to the Emerging World of Polyglot Persistence**. Addison-Wesley Professional, 2013. ISBN 978-0-321-82662-6. Citado nas pp. 22–24.

SCIENCEDIRECT. **ScienceDirect**. 2023. Disponível em: <https://www.sciencedirect.com/>. Acesso em: 4 fev. 2023. Citado na p. 29.

SEBASTIAN, J.; AELTERMAN, S. In: THE Art of SQL Server FILESTREAM. Redgate Books, 2012. Disponível em: <https://www.red-gate.com/library/the-art-of-sql-server-filestream>. Acesso em: 12 fev. 2023. eBook Edition. Citado na p. 18.

SOLARWINDS. **Database Performance Monitoring Software**. Disponível em: <https://www.solarwinds.com/pt/database-performance-monitoring-software>. Acesso em: 5 fev. 2023. Citado na p. 35.

SOLR. **Solr**. Disponível em: <https://solr.apache.org/>. Acesso em: 7 fev. 2023. Citado na p. 30.

SOLR. **Solr Features**. Disponível em: <https://solr.apache.org/features.html>. Acesso em: 8 mai. 2023. Citado na p. 34.

SPICEWORKS. **What Is Cassandra? Meaning, Working, Features, and Uses**. Disponível em: <https://www.spiceworks.com/tech/big-data/articles/what-is-cassandra/>. Acesso em: 7 mai. 2023. Citado na p. 32.

STACK OVERFLOW. **Stack Overflow 2022 Developer Survey**. 2022. Disponível em: <https://survey.stackoverflow.co/2022/#section-most-loved-dreaded-and-wanted-databases>. Acesso em: 7 mai. 2023. Citado nas pp. 29, 31–33.

STATISTA. **Ranking of the most popular database management systems worldwide, as of August 2022**. 2022. Disponível em: <https://www.statista.com/statistics/809750/worldwide-popularity-ranking-database-management-systems/>. Acesso em: 4 fev. 2023. Citado na p. 29.

XIAO, Z.; LIU, Y. Remote sensing image database based on NOSQL database. In: 2011 19th International Conference on Geoinformatics. 2011. P. 1–5. DOI: 10.1109/GeoInformatics.2011.5980724. Citado na p. 26.

Apêndice A – Gráficos do tipo violino para análise da distribuição dos tempos de resposta

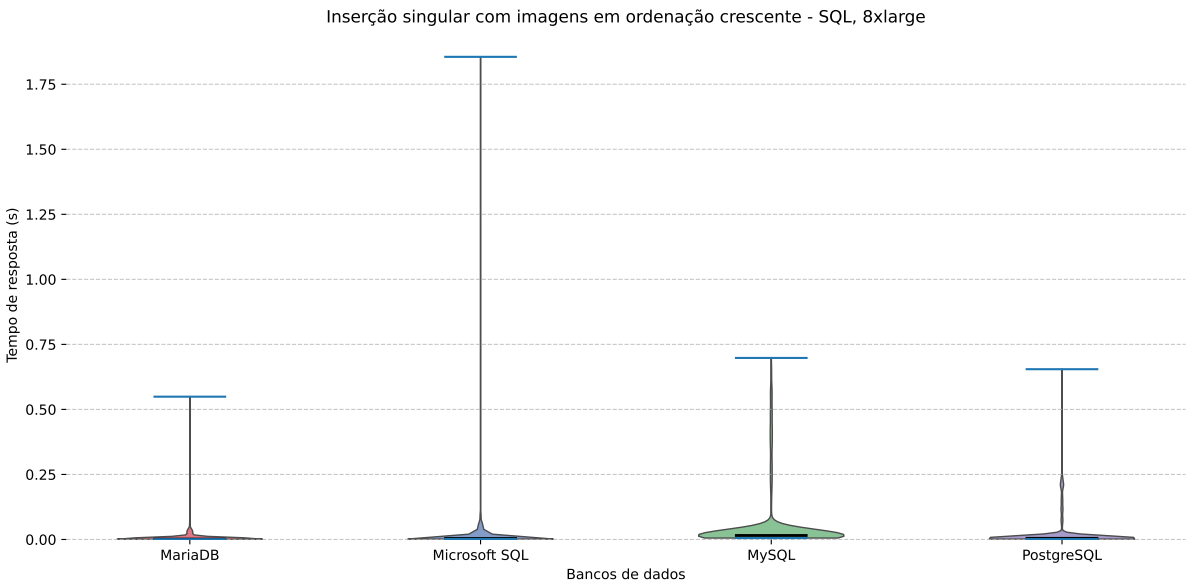

Figura A.46 – Gráfico dos tempos de resposta (inserção única, crescente, SQL, 8xlarge)

Fonte: Produzido pelos autores.

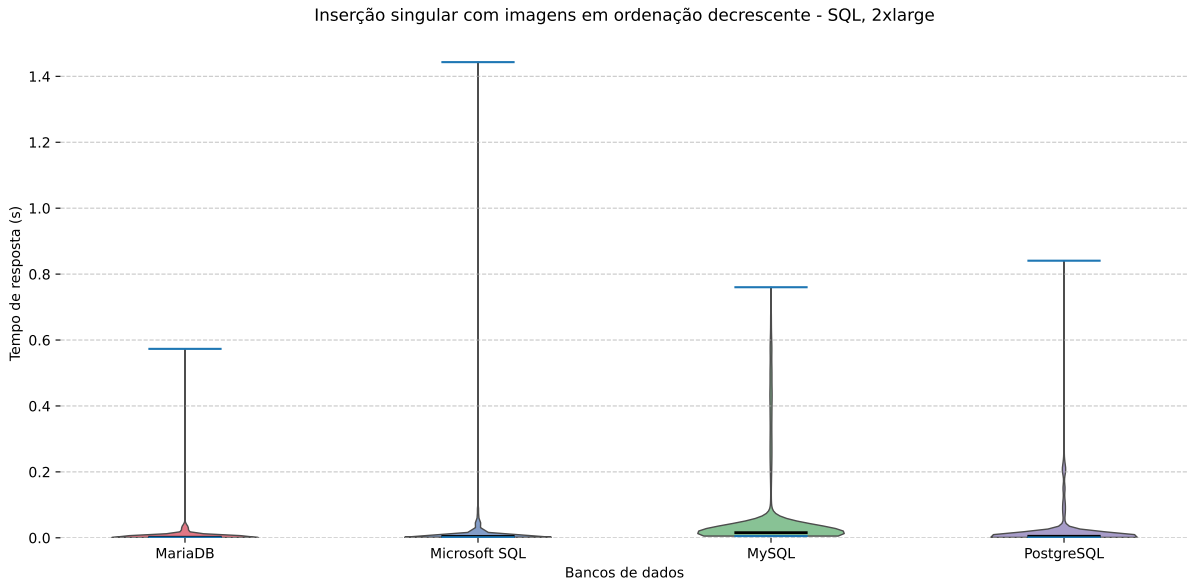

Figura A.47 – Gráfico dos tempos de resposta (inserção única, decrescente, SQL, 2xlarge)

Fonte: Produzido pelos autores.

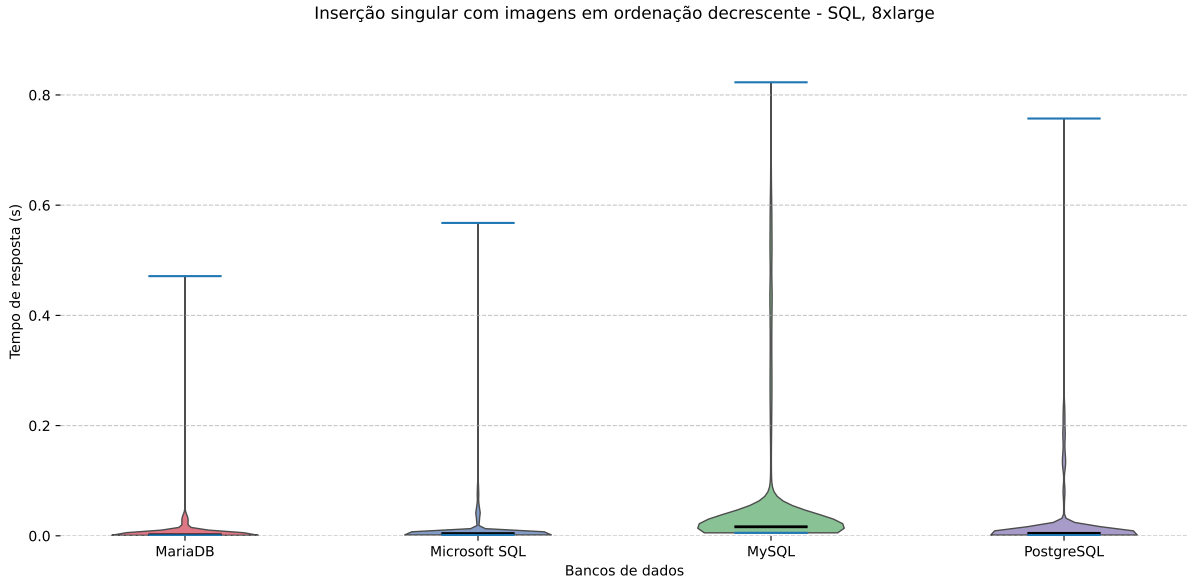

Figura A.48 – Gráfico dos tempos de resposta (inserção única, decrescente, SQL, 8xlarge)

Fonte: Produzido pelos autores.

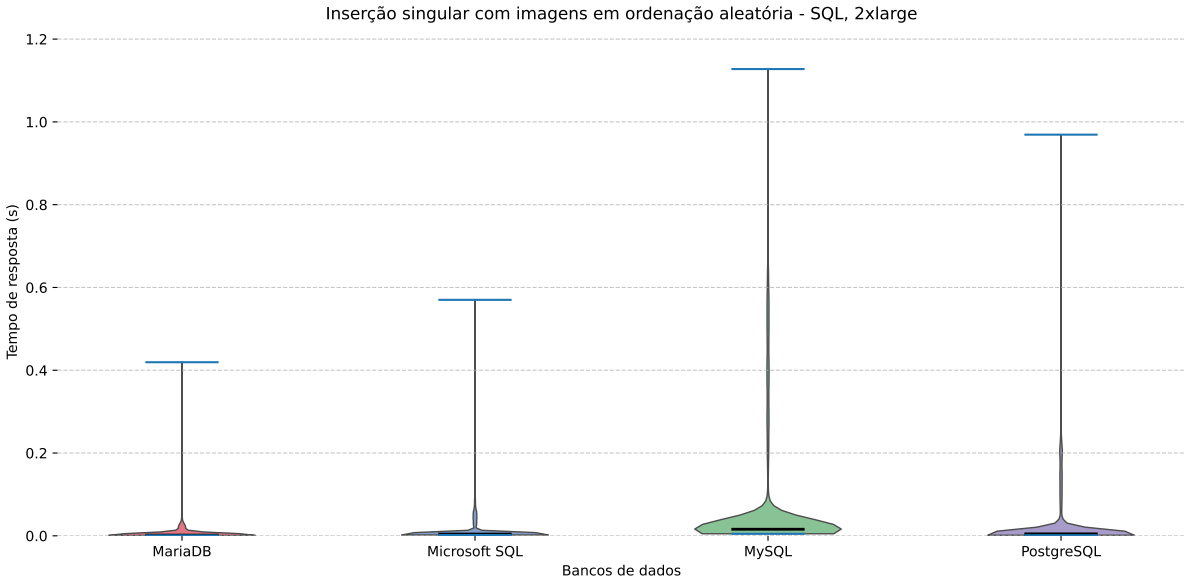

Figura A.49 – Gráfico dos tempos de resposta (inserção única, aleatória, SQL, 2xlarge)

Fonte: Produzido pelos autores.

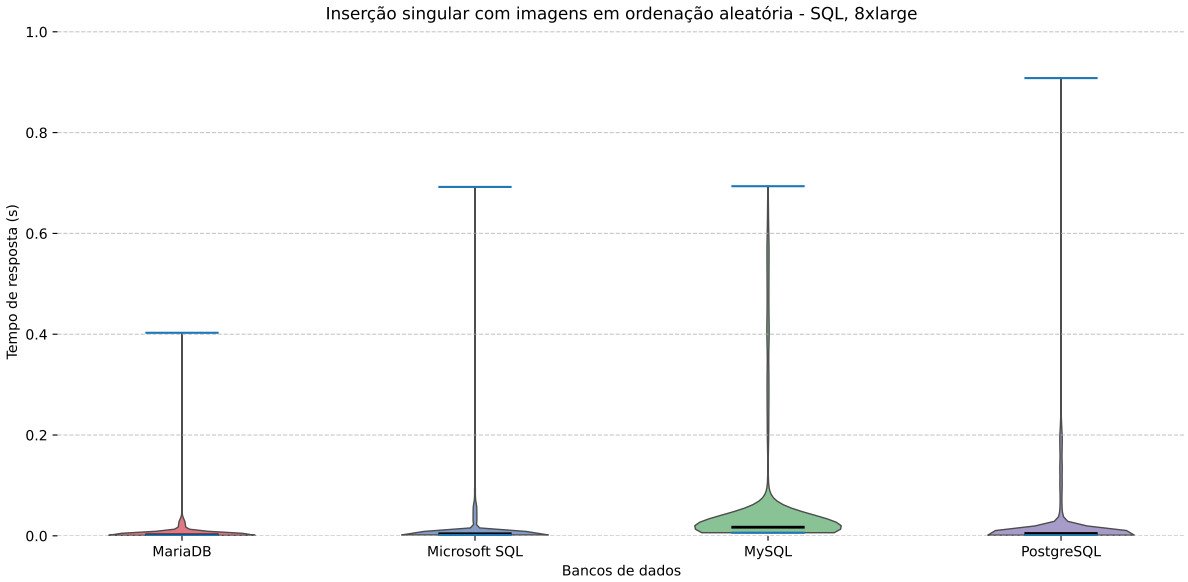

Figura A.50 – Gráfico dos tempos de resposta (inserção única, aleatória, SQL, 8xlarge)

Fonte: Produzido pelos autores.

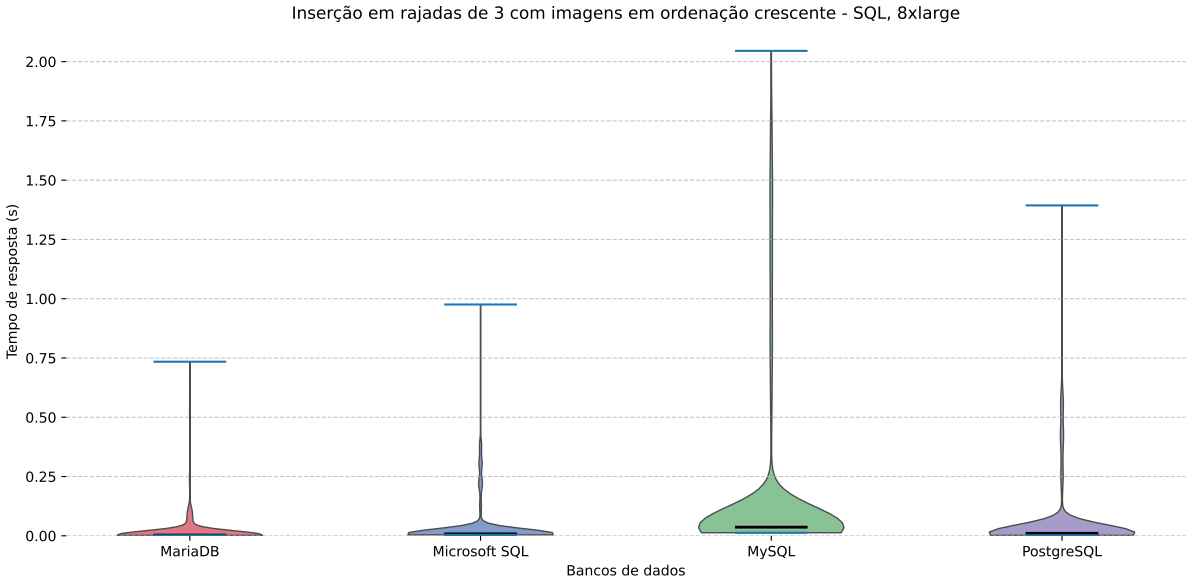

Figura A.51 – Gráfico dos tempos de resposta (inserção em rajadas, crescente, SQL, 8xlarge)

Fonte: Produzido pelos autores.

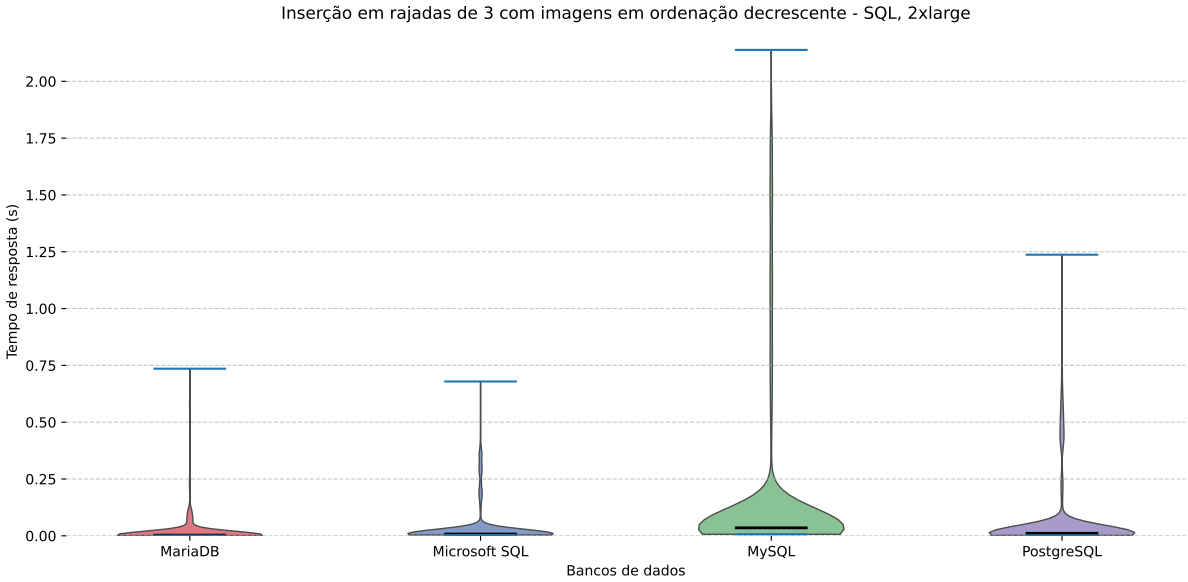

Figura A.52 – Gráfico dos tempos de resposta (inserção em rajadas, decrescente, SQL, 2xlarge)

Fonte: Produzido pelos autores.

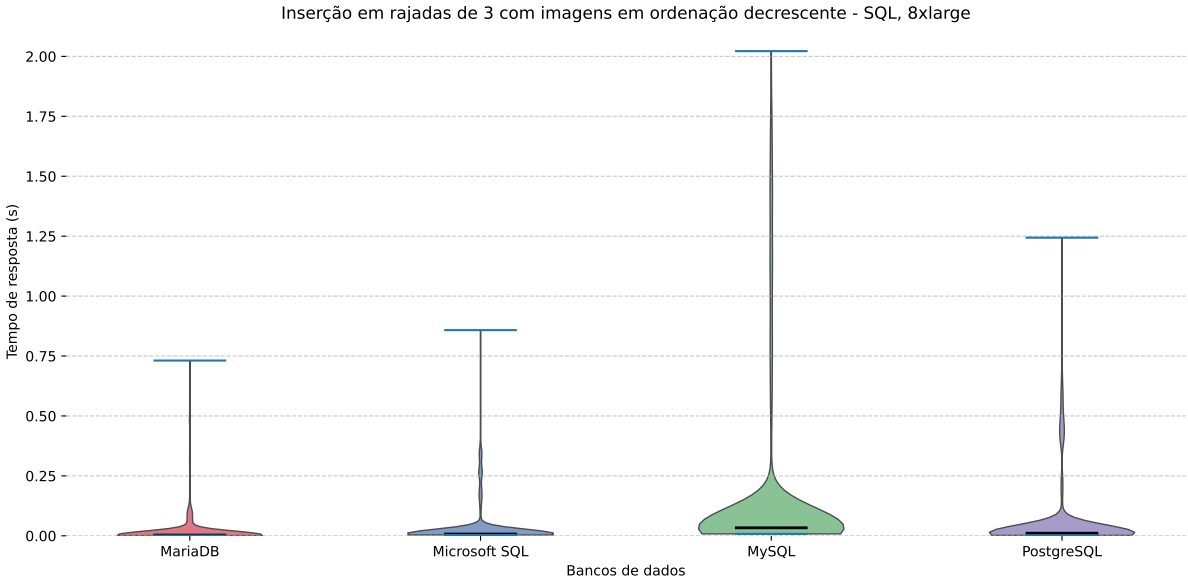

Figura A.53 – Gráfico dos tempos de resposta (inserção em rajadas, decrescente, SQL, 8xlarge)

Fonte: Produzido pelos autores.

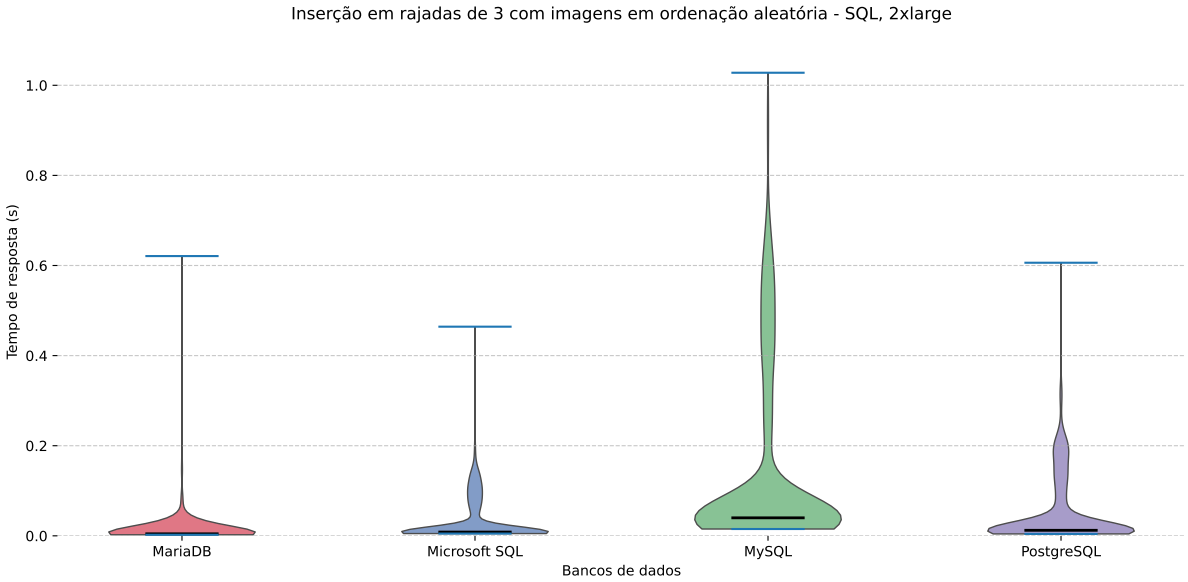

Figura A.54 – Gráfico dos tempos de resposta (inserção em rajadas, aleatória, SQL, 2xlarge)

Fonte: Produzido pelos autores.

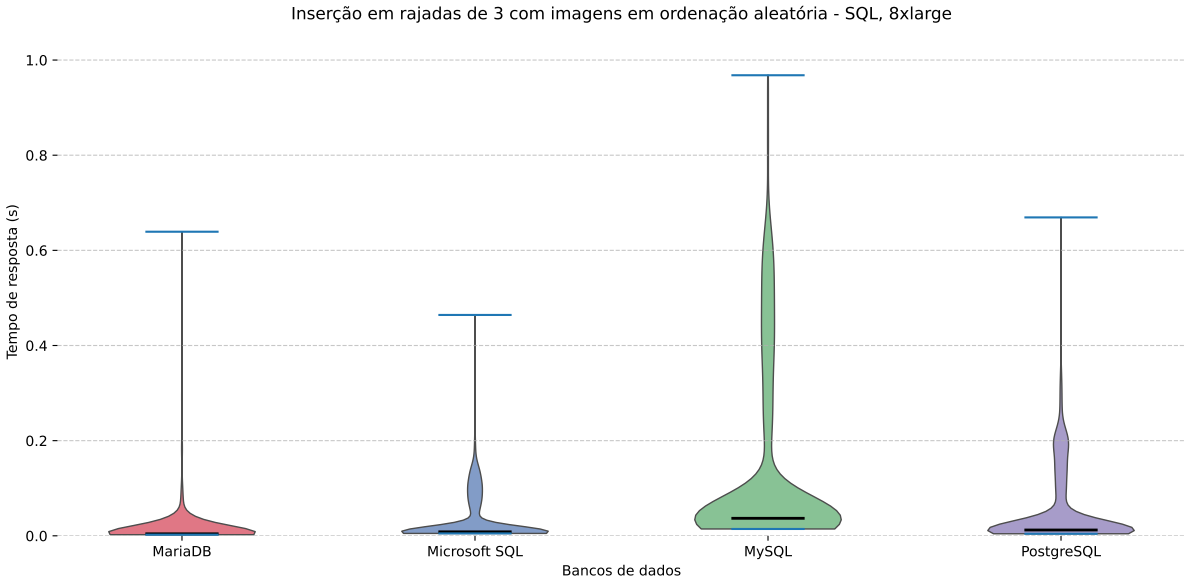

Figura A.55 – Gráfico dos tempos de resposta (inserção em rajadas, aleatória, SQL, 8xlarge)

Fonte: Produzido pelos autores.

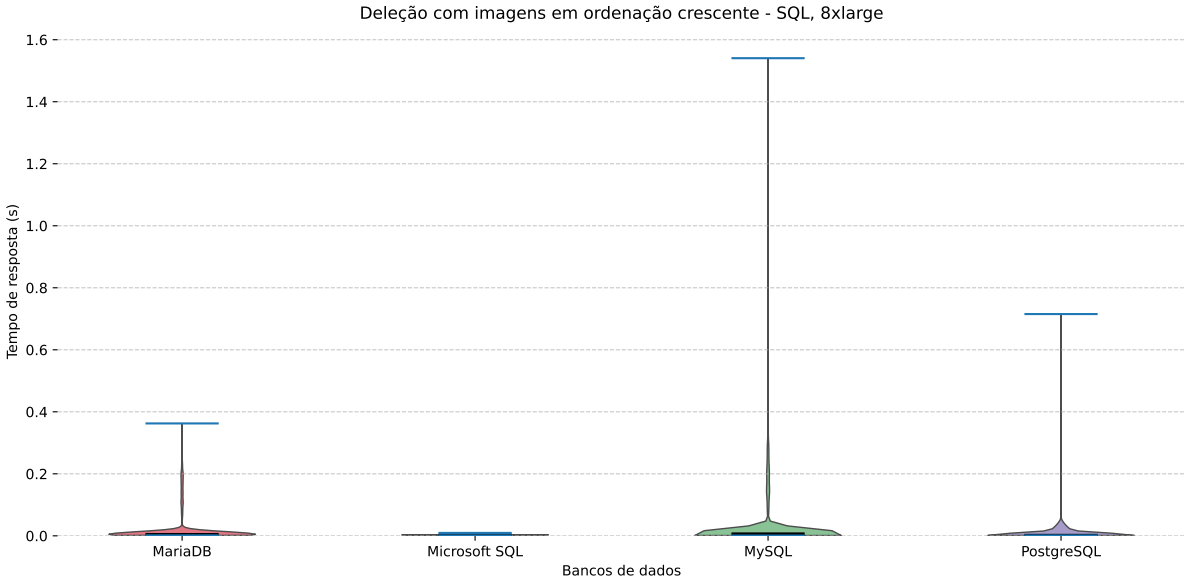

Figura A.56 – Gráfico dos tempos de resposta (deleção, crescente, SQL, 8xlarge)

Fonte: Produzido pelos autores.

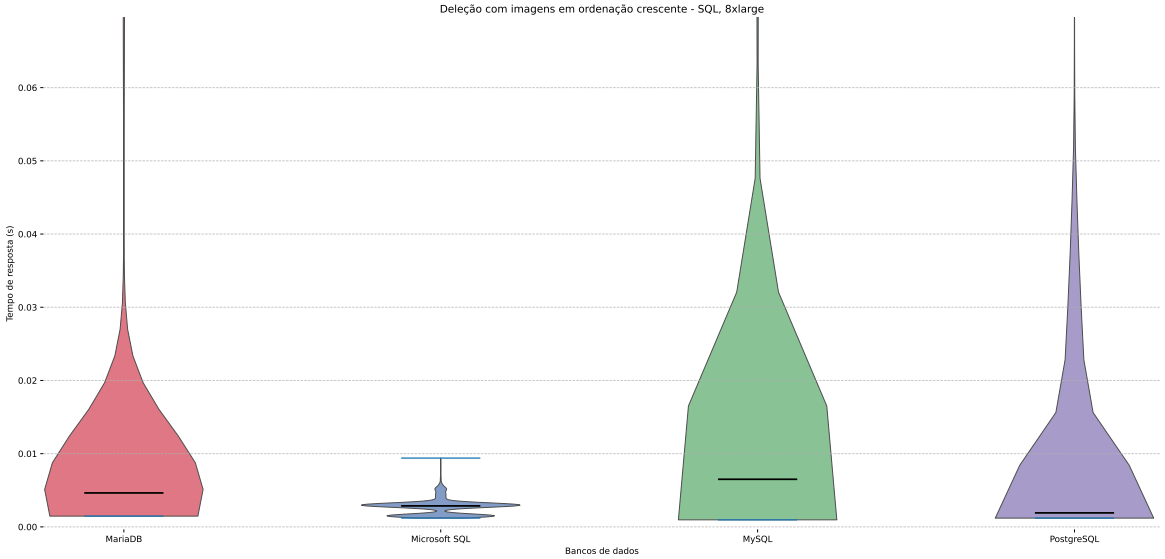

Figura A.57 – *Zoom* para inspeção da curva do Microsoft SQL da Figura A.56

Fonte: Produzido pelos autores.

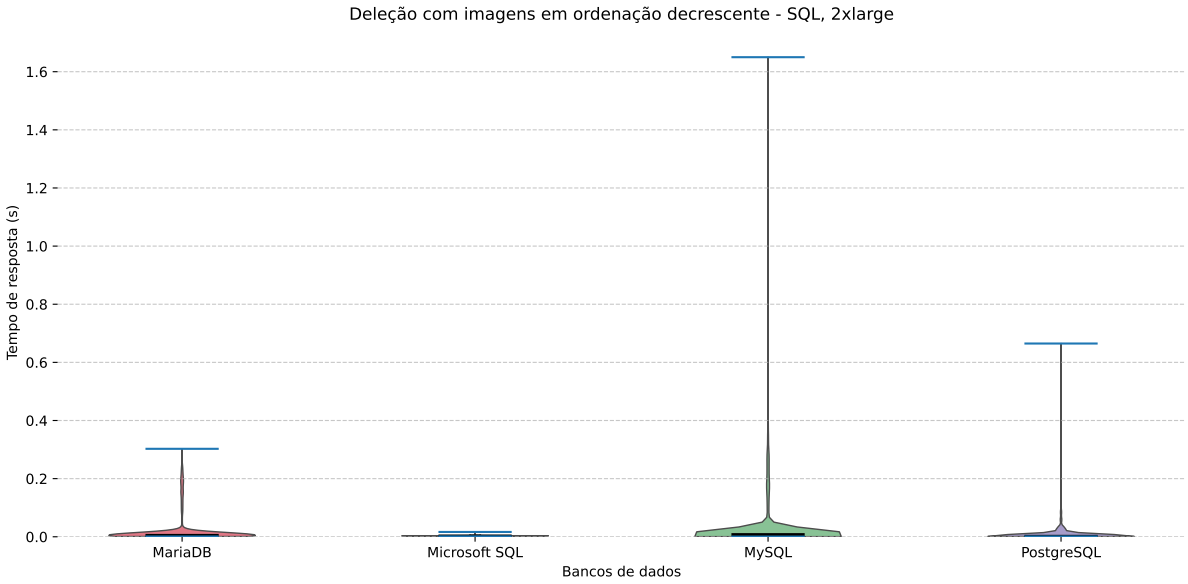

Figura A.58 – Gráfico dos tempos de resposta (deleção, decrescente, SQL, 2xlarge)

Fonte: Produzido pelos autores.

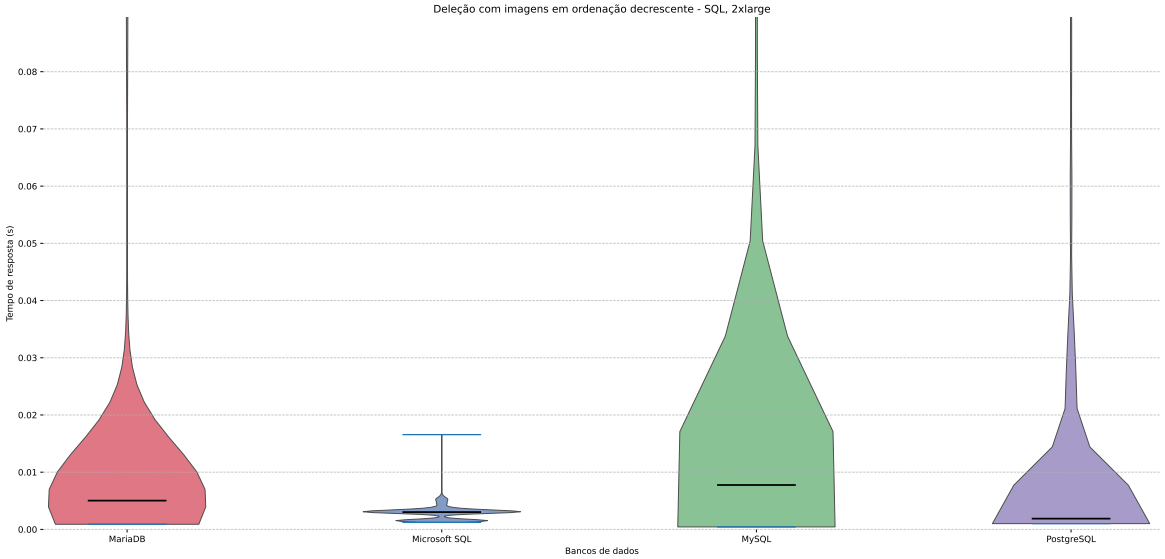

Figura A.59 – *Zoom* para inspeção da curva do Microsoft SQL da Figura A.58

Fonte: Produzido pelos autores.

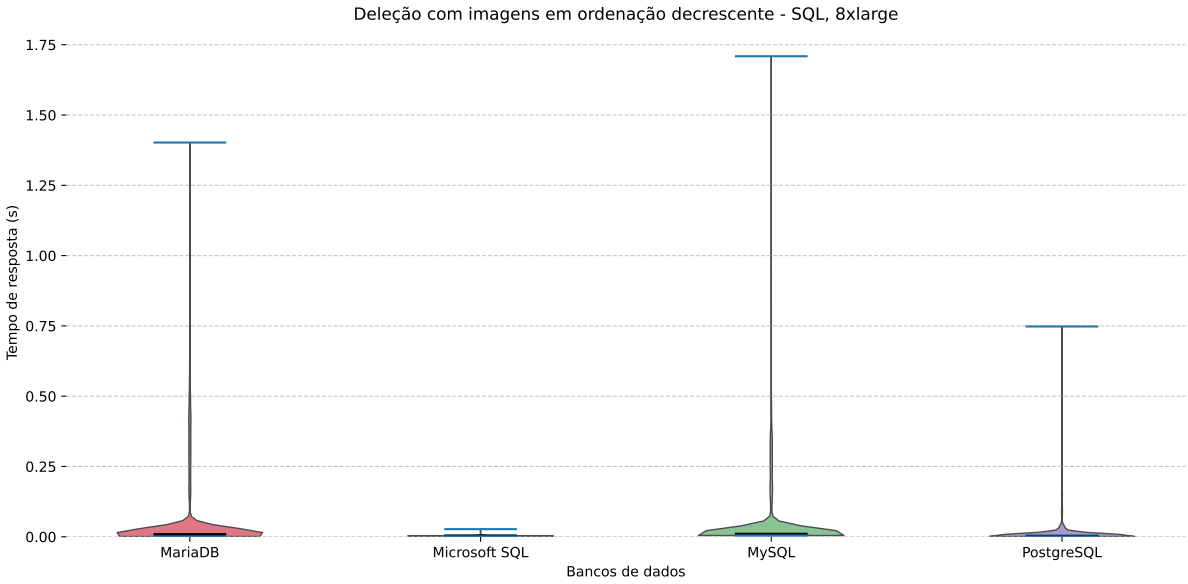

Figura A.60 – Gráfico dos tempos de resposta (deleção, derescente, SQL, 8xlarge)

Fonte: Produzido pelos autores.

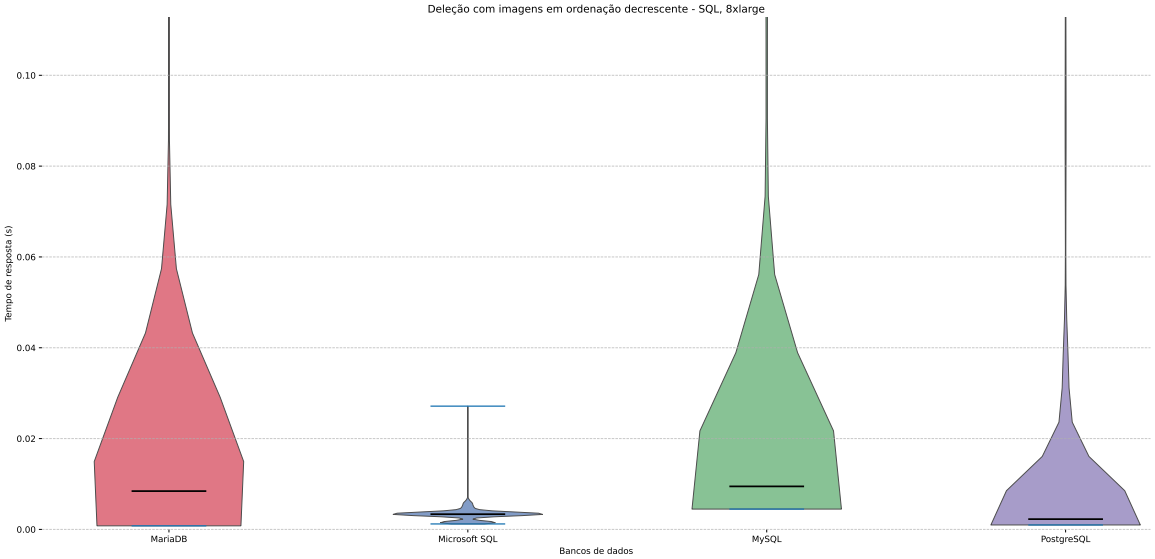

Figura A.61 – *Zoom* para inspeção da curva do Microsoft SQL da Figura A.60

Fonte: Produzido pelos autores.

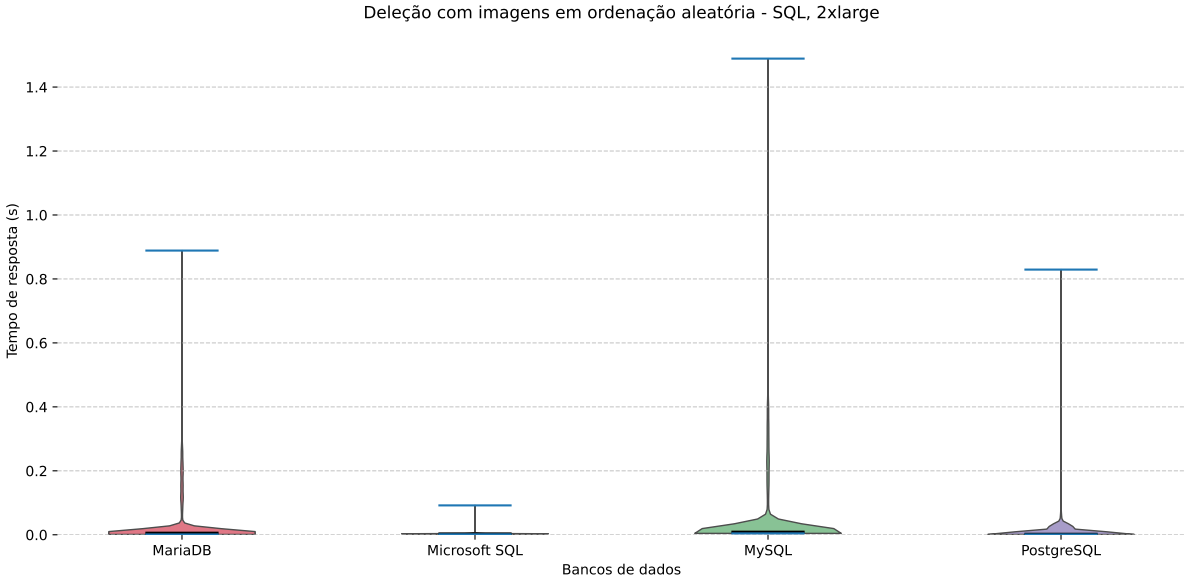

Figura A.62 – Gráfico dos tempos de resposta (deleção, aleatória, SQL, 2xlarge)

Fonte: Produzido pelos autores.

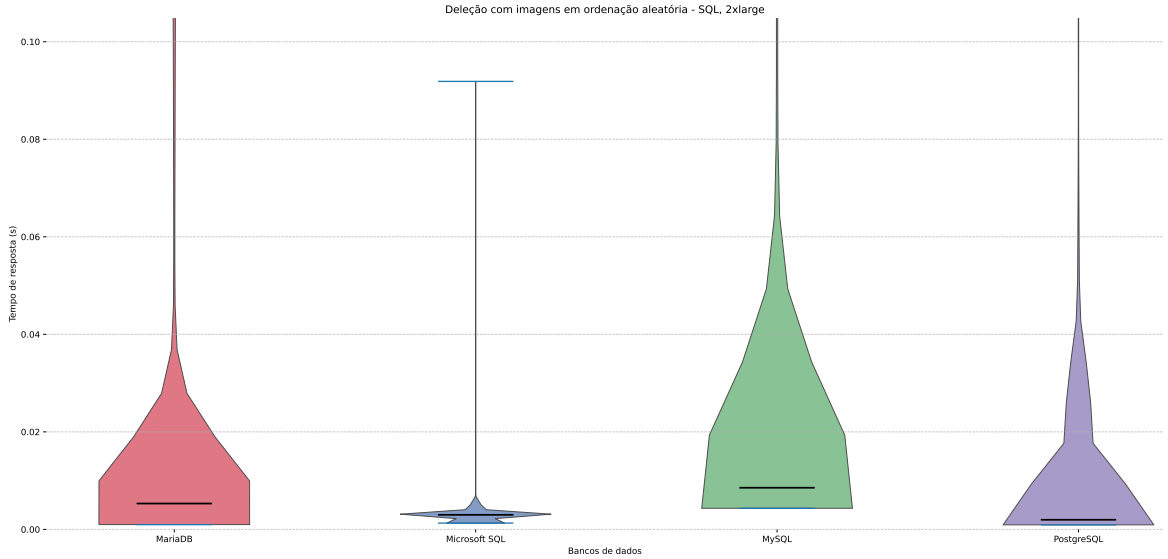

Figura A.63 – *Zoom* para inspeção da curva do Microsoft SQL da Figura A.62

Fonte: Produzido pelos autores.

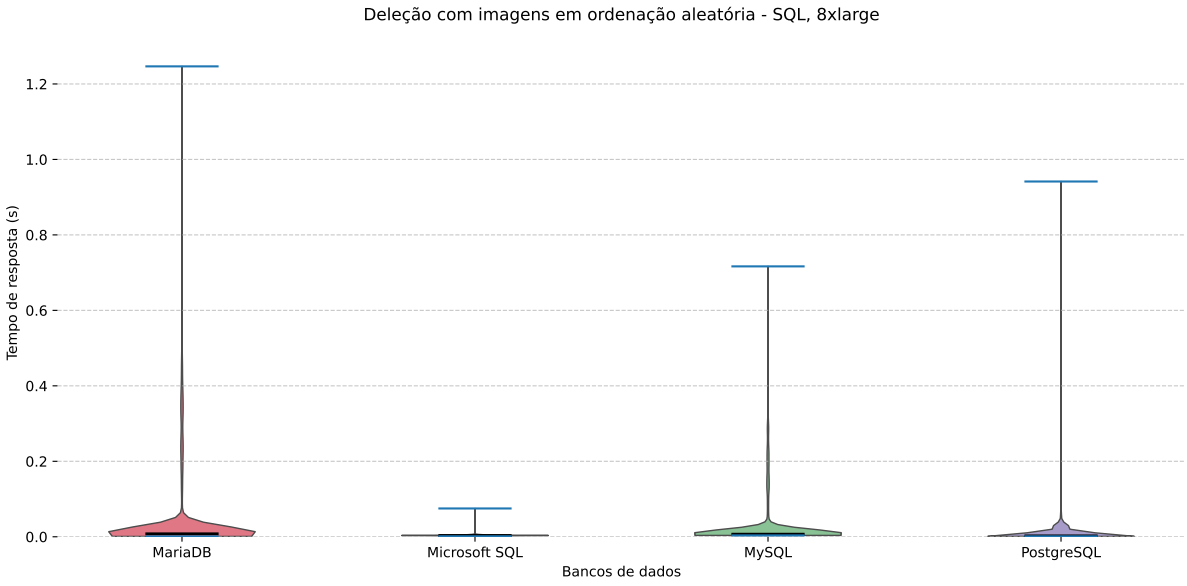

Figura A.64 – Gráfico dos tempos de resposta (deleção, aleatória, SQL, 8xlarge)

Fonte: Produzido pelos autores.

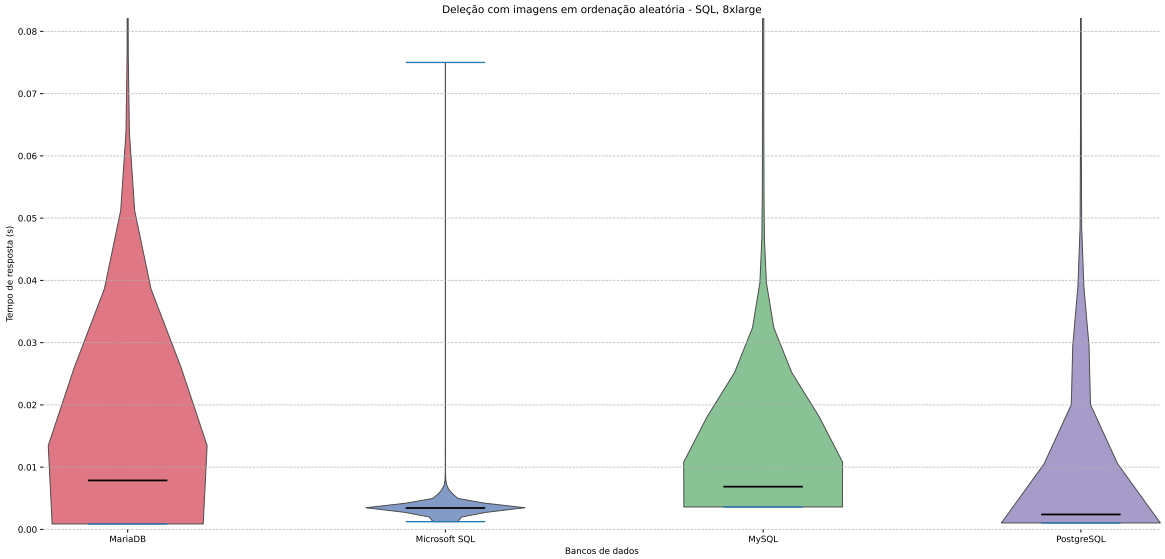

Figura A.65 – *Zoom* para inspeção da curva do Microsoft SQL da Figura A.64

Fonte: Produzido pelos autores.

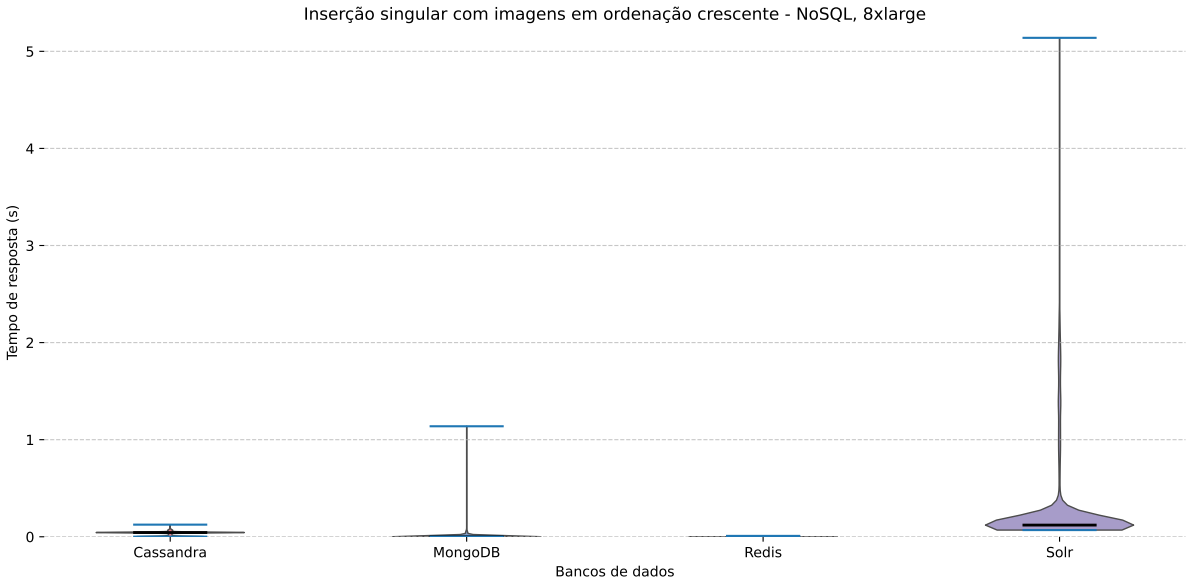

Figura A.66 – Gráfico dos tempos de resposta (inserção única, crescente, NoSQL, 8xlarge)

Fonte: Produzido pelos autores.

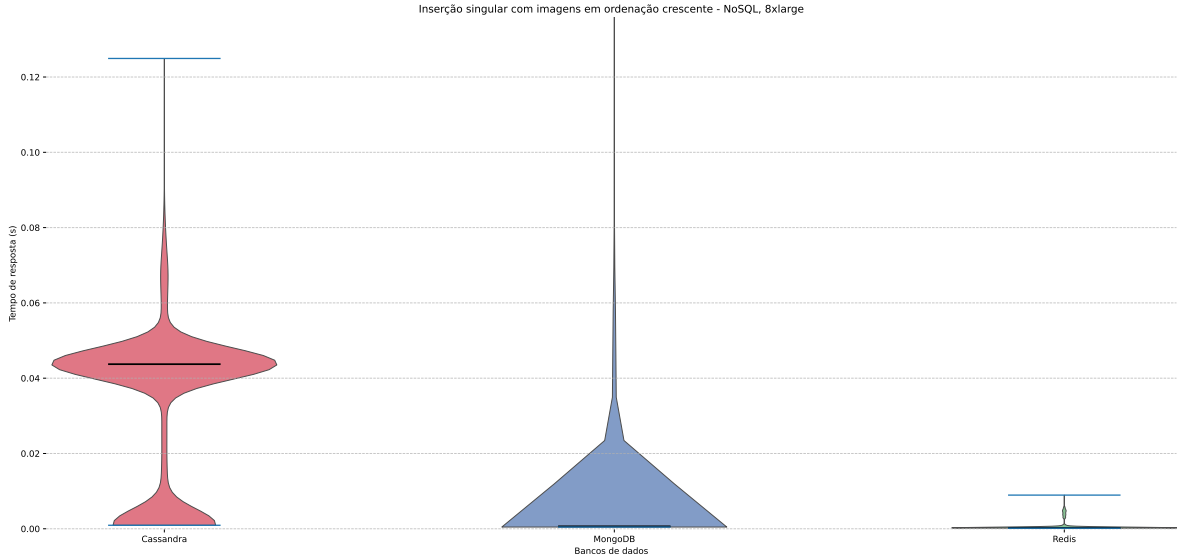

Figura A.67 – *Zoom* para inspeção das curvas do Cassandra e Redis da Figura A.66

Fonte: Produzido pelos autores.

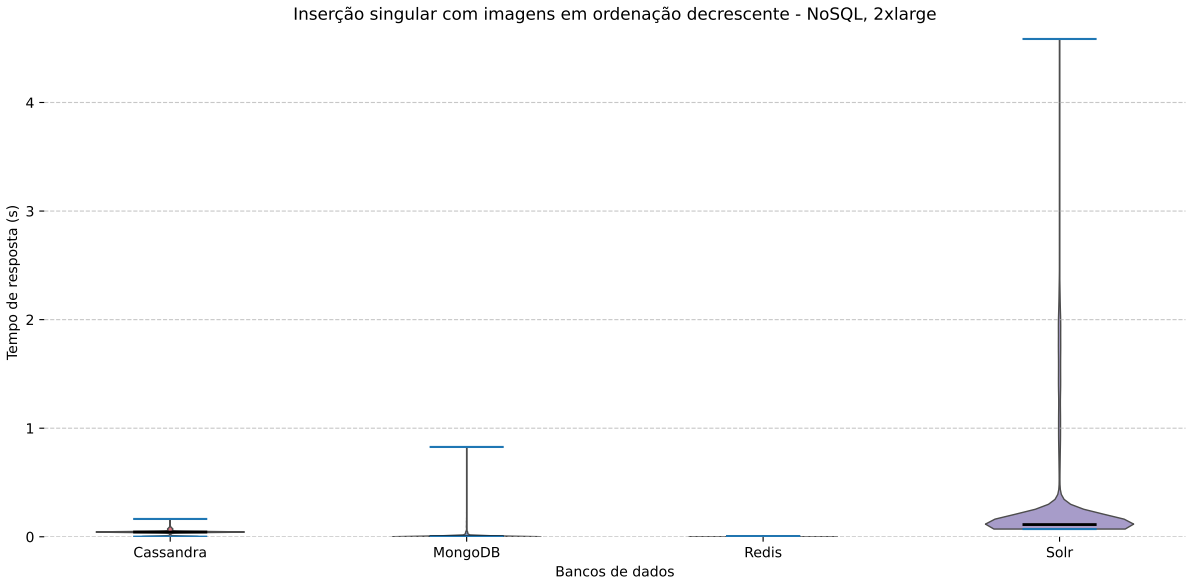

Figura A.68 – Gráfico dos tempos de resposta (inserção única, decrescente, NoSQL, 2xlarge)

Fonte: Produzido pelos autores.

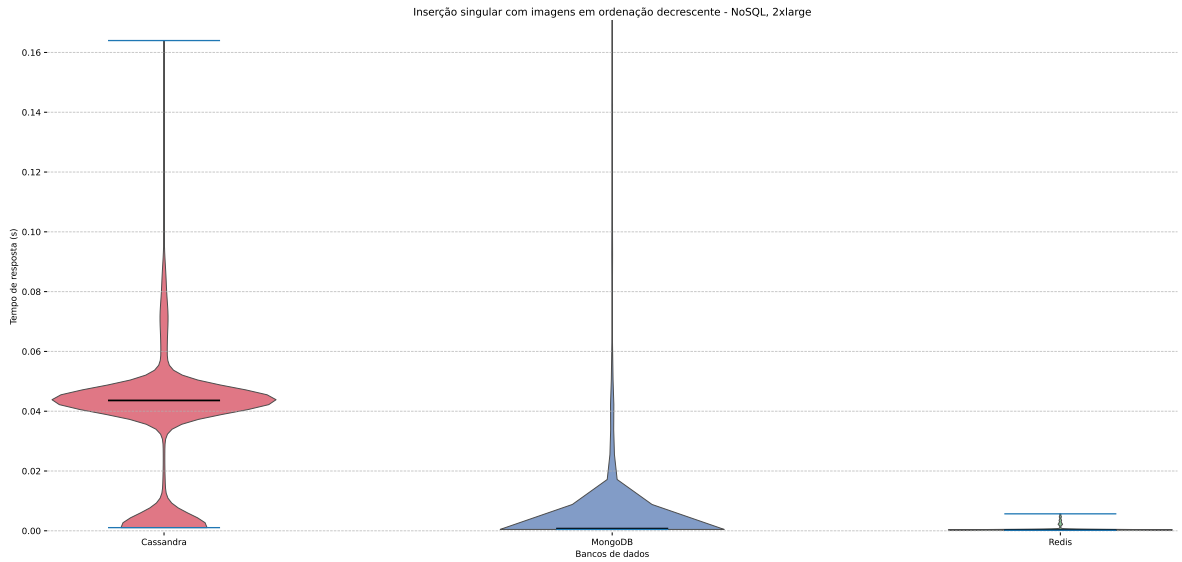

Figura A.69 – *Zoom* para inspeção das curvas do Cassandra e Redis da Figura A.68

Fonte: Produzido pelos autores.

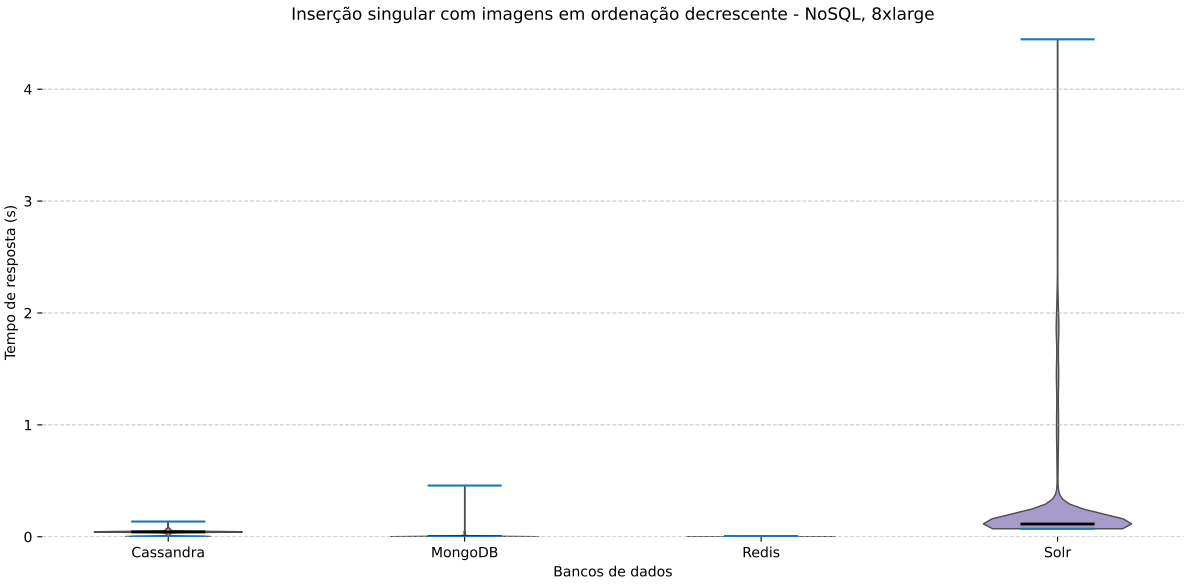

Figura A.70 – Gráfico dos tempos de resposta (inserção única, decrescente, NoSQL, 8xlarge)

Fonte: Produzido pelos autores.

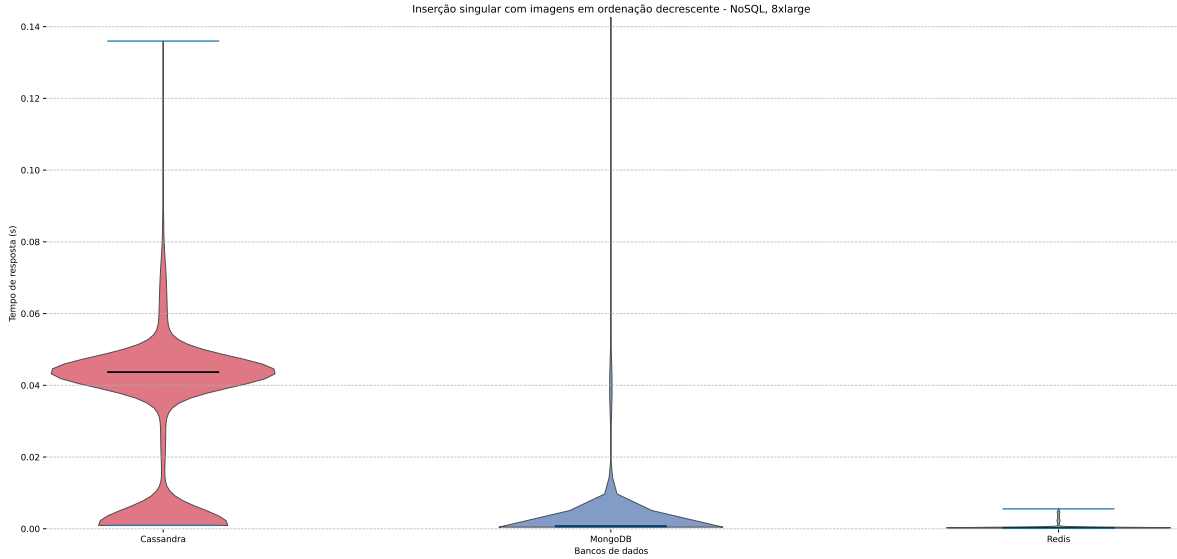

Figura A.71 – *Zoom* para inspeção das curvas do Cassandra e Redis da Figura A.70

Fonte: Produzido pelos autores.

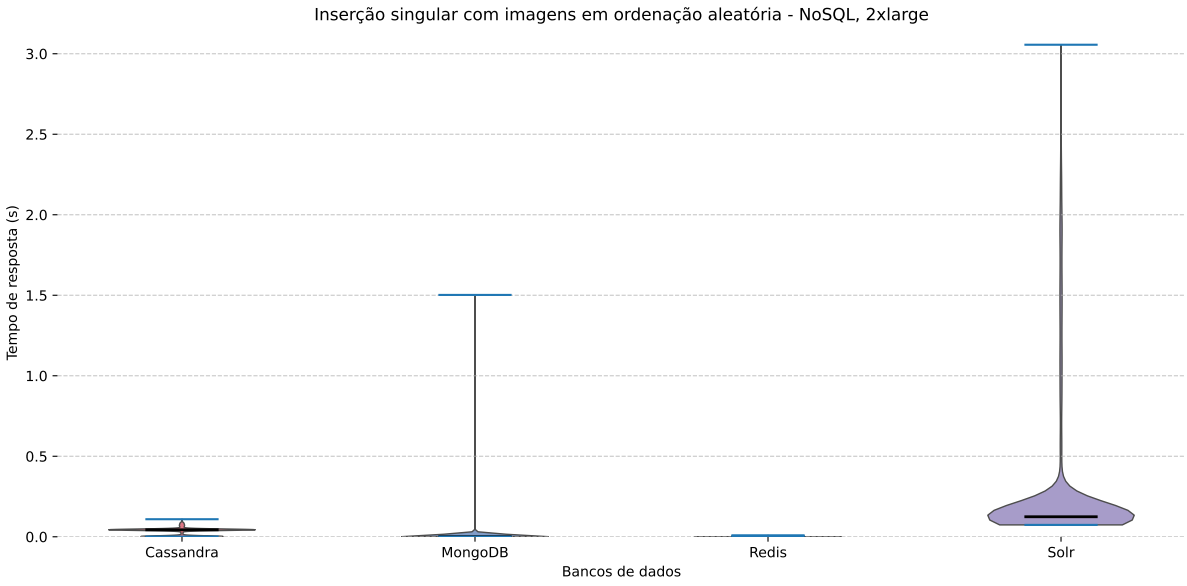

Figura A.72 – Gráfico dos tempos de resposta (inserção única, aleatória, NoSQL, 2xlarge)

Fonte: Produzido pelos autores.

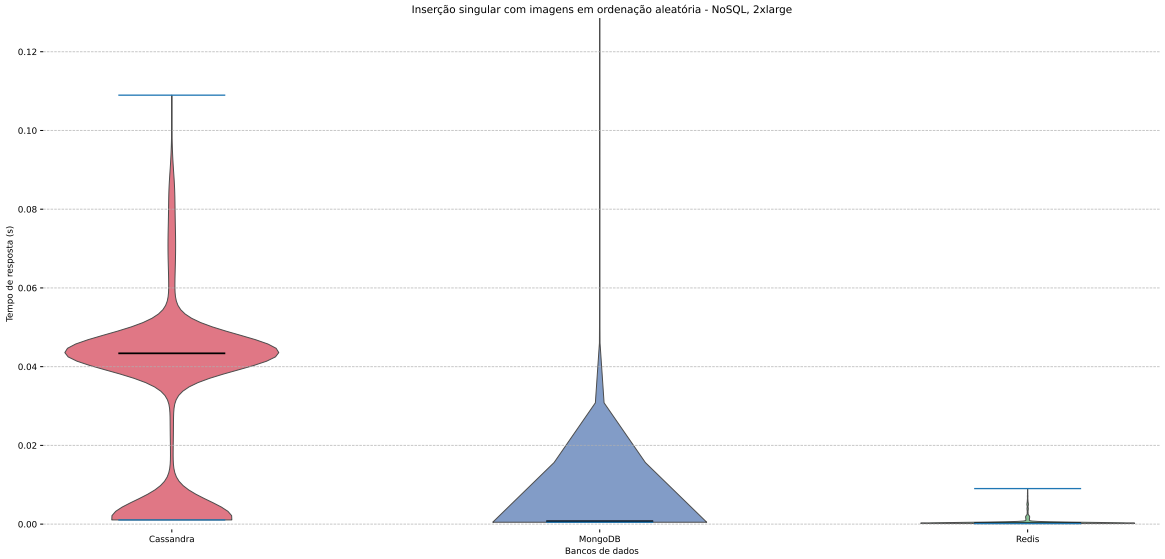

Figura A.73 – *Zoom* para inspeção das curvas do Cassandra e Redis da Figura A.72

Fonte: Produzido pelos autores.

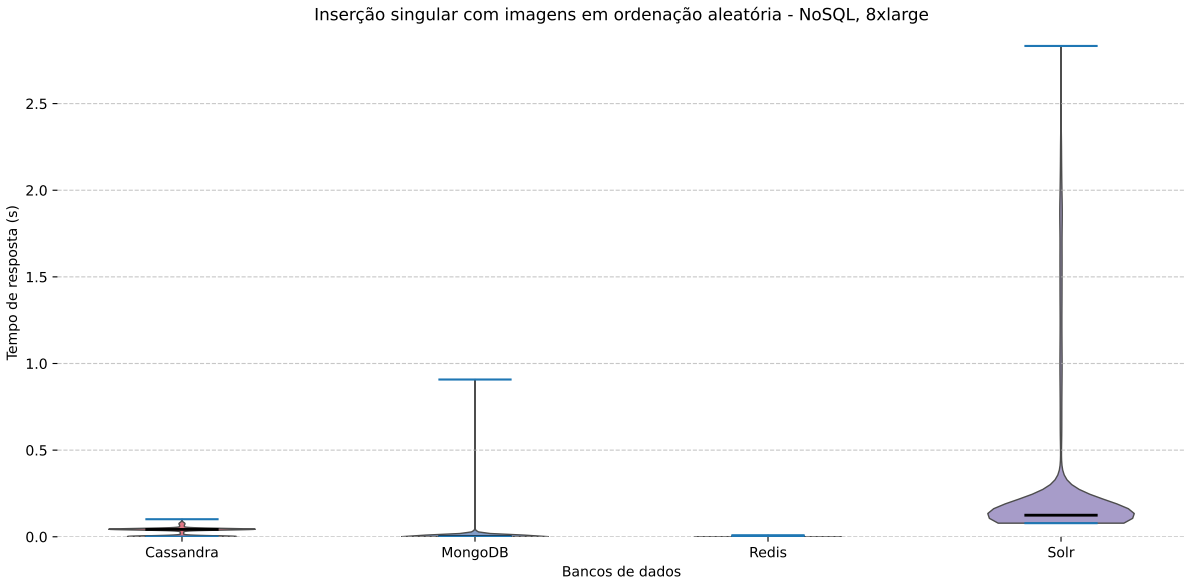

Figura A.74 – Gráfico dos tempos de resposta (inserção única, aleatória, NoSQL, 8xlarge)

Fonte: Produzido pelos autores.

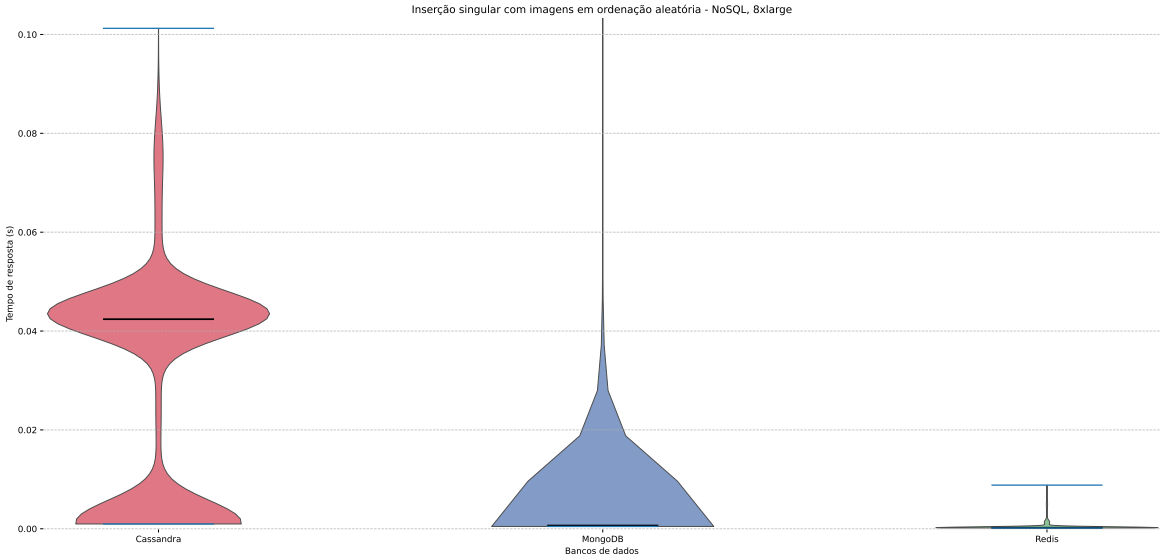

Figura A.75 – *Zoom* para inspeção das curvas do Cassandra e Redis da Figura A.74

Fonte: Produzido pelos autores.

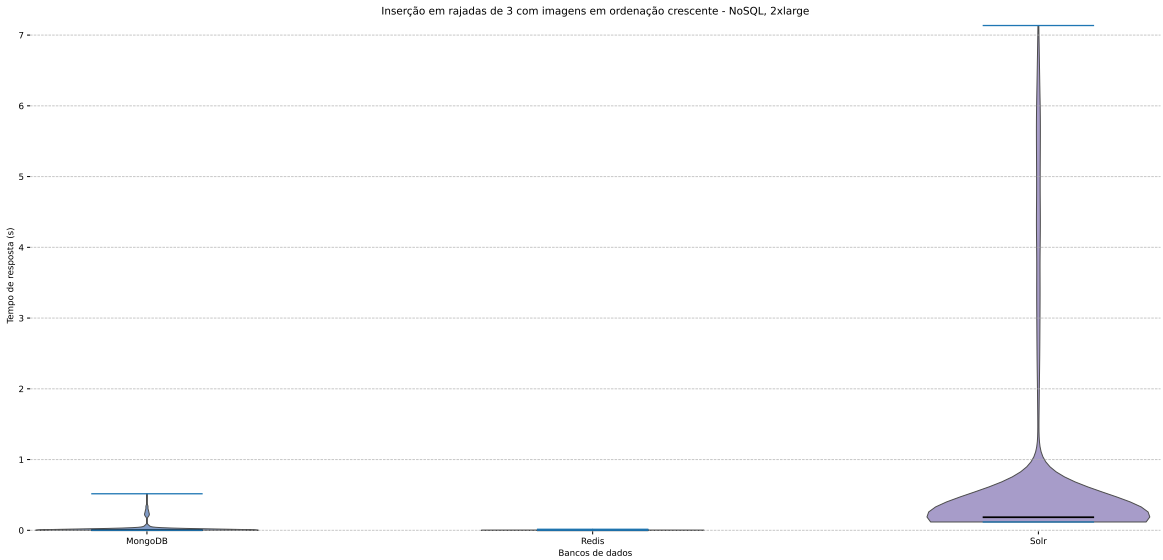

Figura A.76 – Gráfico dos tempos de resposta (inserção em rajadas, crescente, NoSQL, 2xlarge)

Fonte: Produzido pelos autores.

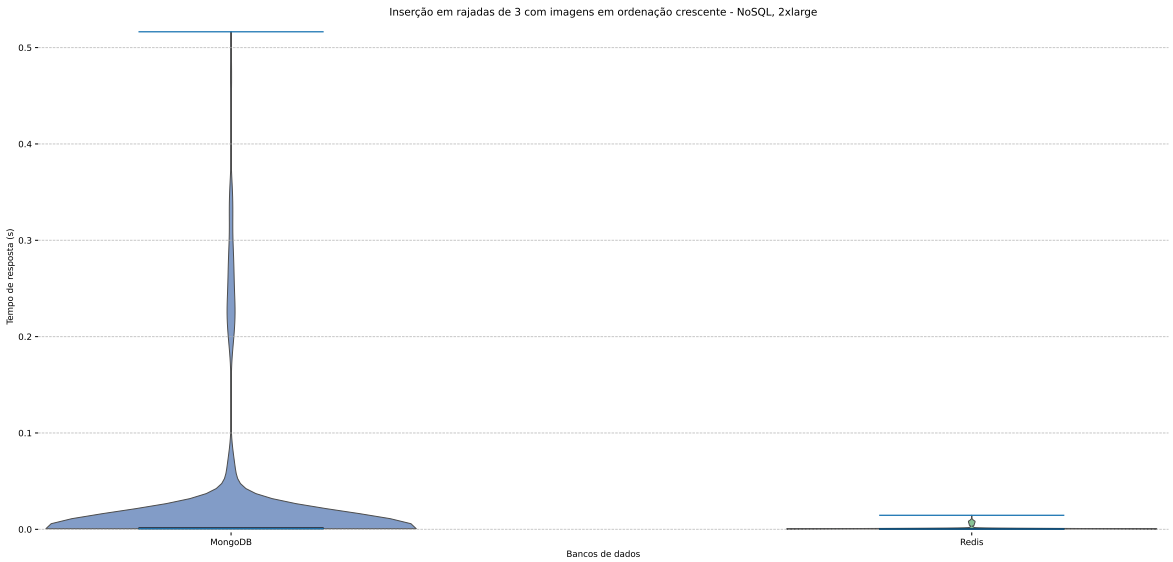

Figura A.77 – *Zoom* para inspeção das curvas do MongoDB e Redis da Figura A.76

Fonte: Produzido pelos autores.

Figura A.78 – Gráfico dos tempos de resposta (inserção em rajadas, crescente, NoSQL, 8xlarge)

Fonte: Produzido pelos autores.

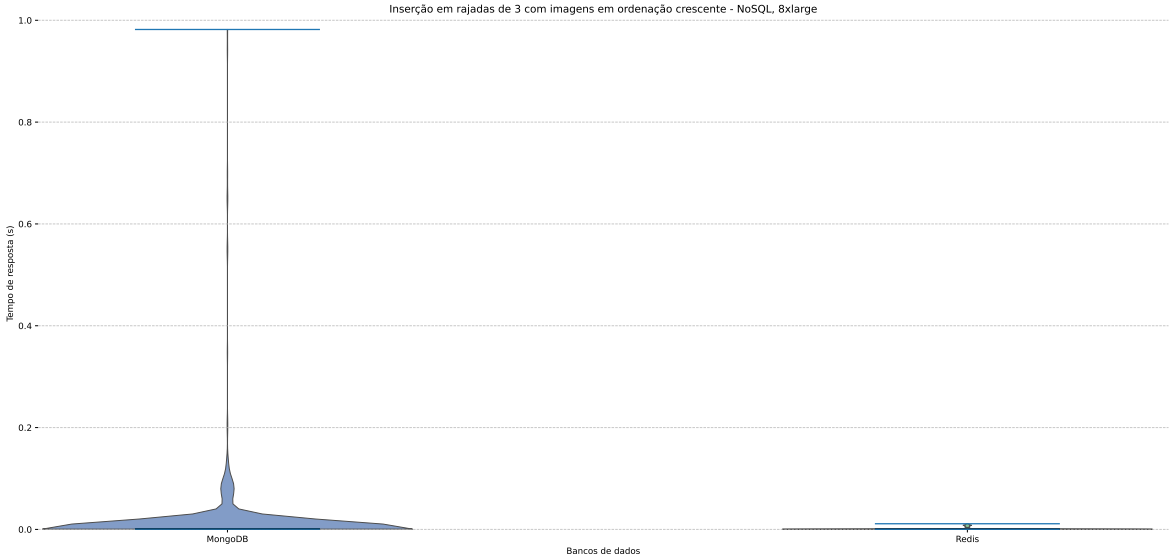

Figura A.79 – *Zoom* para inspeção das curvas do MongoDB e Redis da Figura A.78

Fonte: Produzido pelos autores.

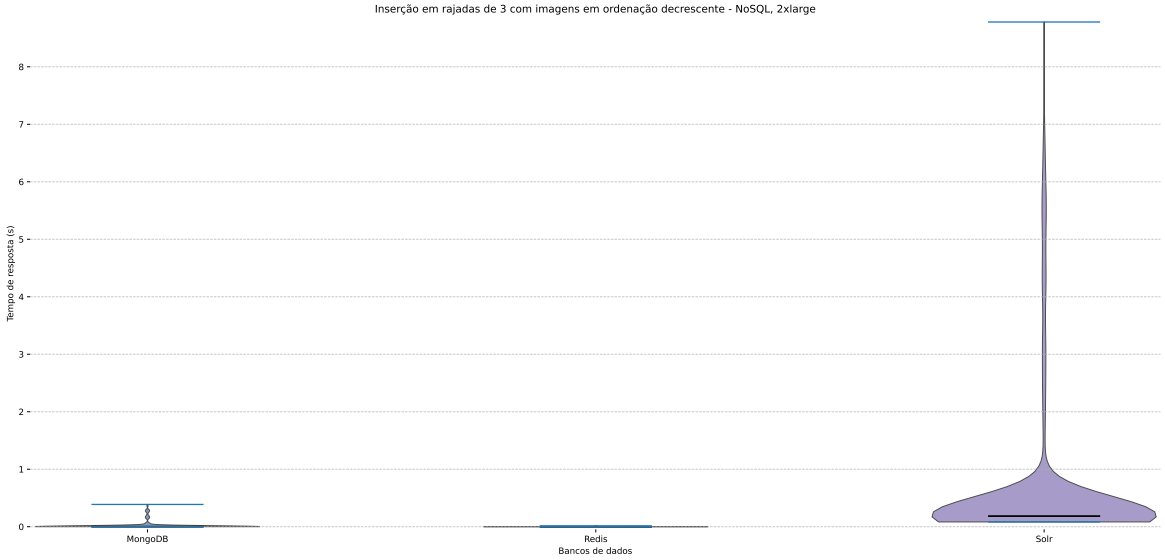

Figura A.80 – Gráfico dos tempos de resposta (inserção em rajadas, decrescente, NoSQL, 2xlarge)

Fonte: Produzido pelos autores.

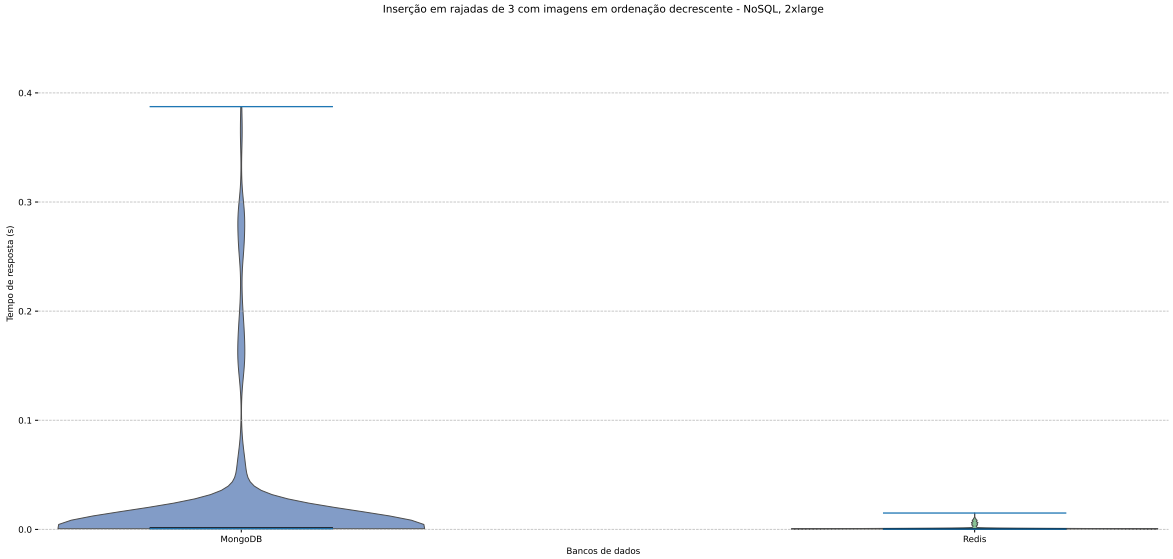

Figura A.81 – *Zoom* para inspeção das curvas do MongoDB e Redis da Figura A.80

Fonte: Produzido pelos autores.

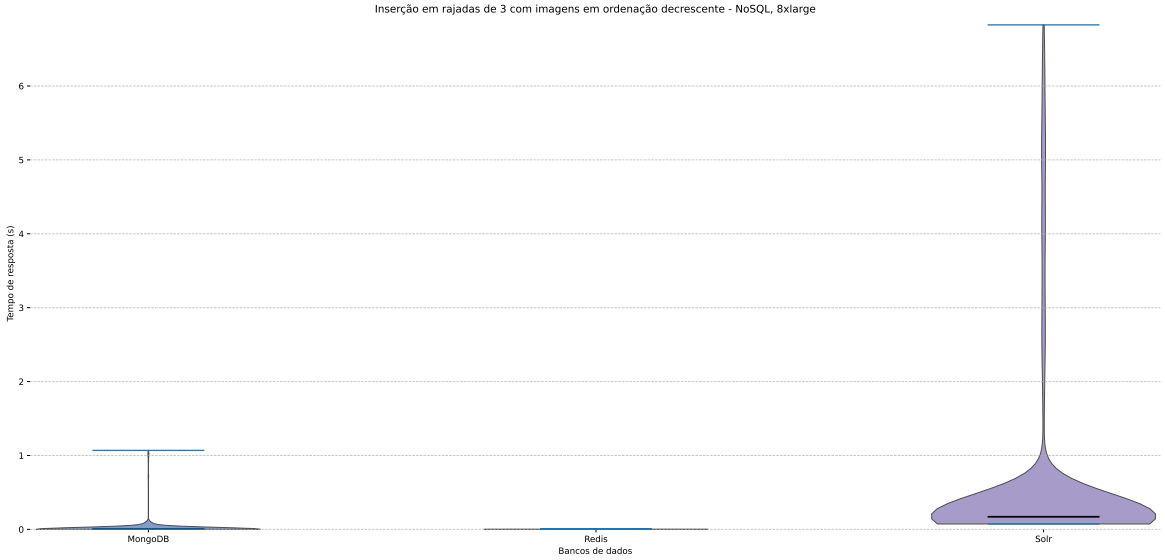

Figura A.82 – Gráfico dos tempos de resposta (inserção em rajadas, decrescente, NoSQL, 8xlarge)

Fonte: Produzido pelos autores.

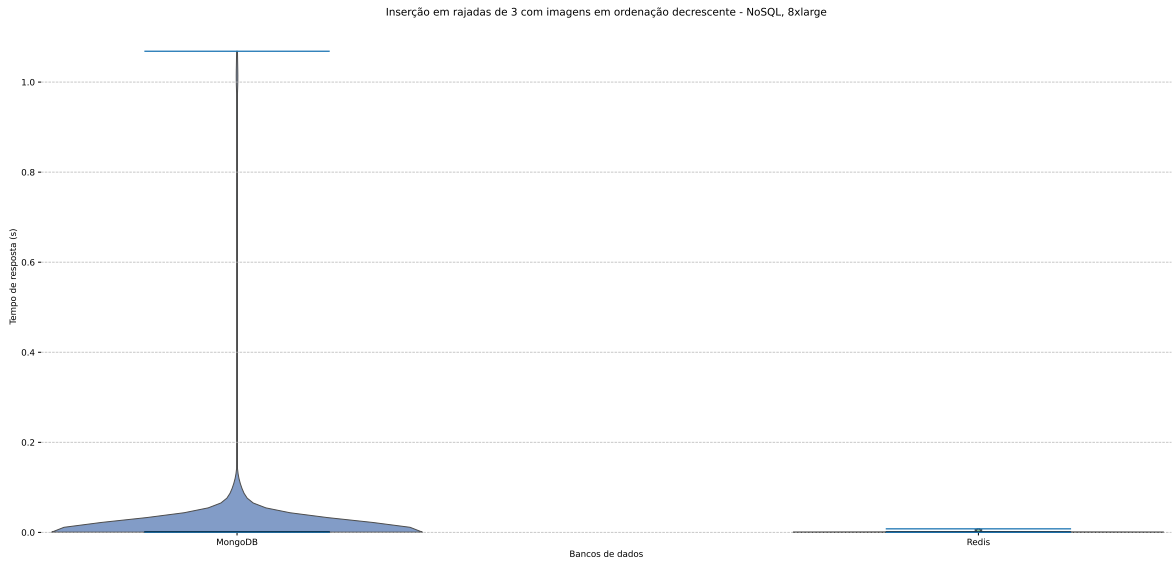

Figura A.83 – *Zoom* para inspeção das curvas do MongoDB e Redis da Figura A.82

Fonte: Produzido pelos autores.

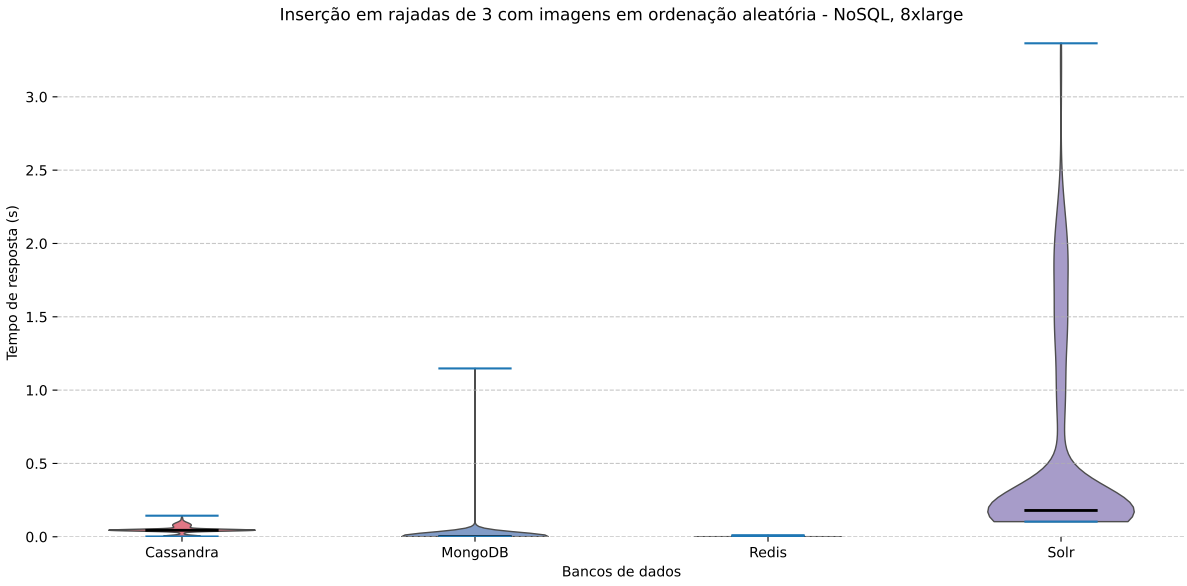

Figura A.84 – Gráfico dos tempos de resposta (inserção em rajadas, aleatória, NoSQL, 8xlarge)

Fonte: Produzido pelos autores.

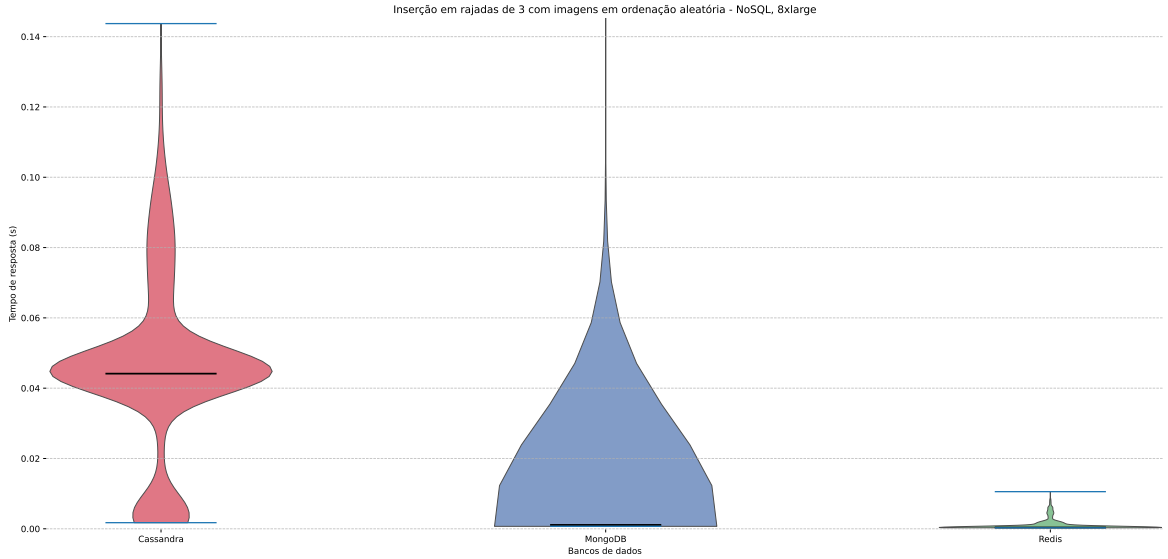

Figura A.85 – *Zoom* para inspeção das curvas do Cassandra e Redis da Figura A.84

Fonte: Produzido pelos autores.

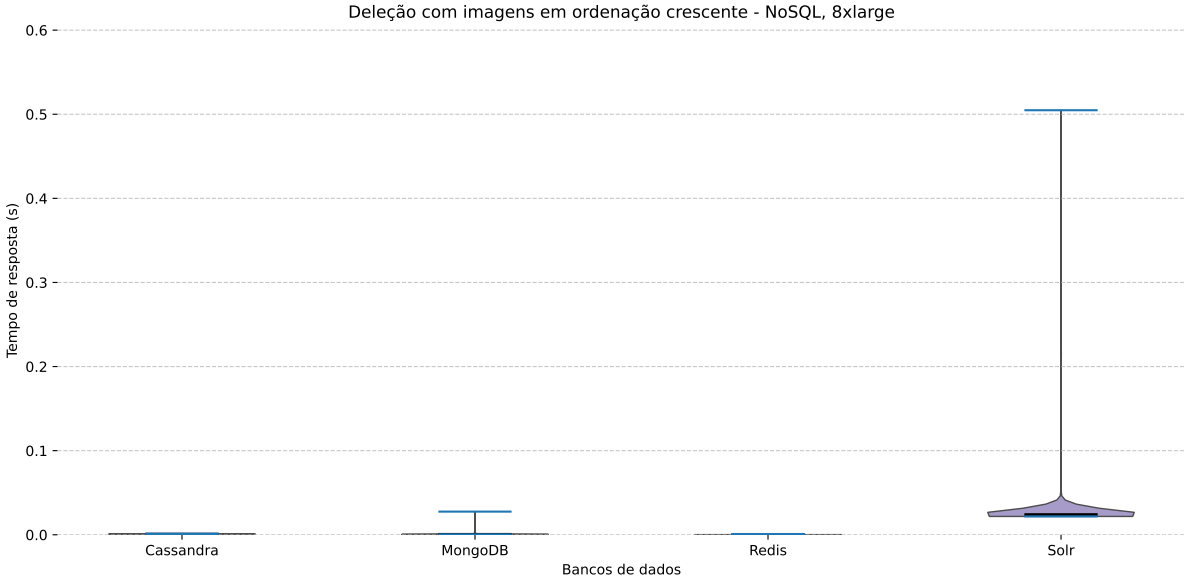

Figura A.86 – Gráfico dos tempos de resposta (deleção, crescente, NoSQL, 8xlarge)

Fonte: Produzido pelos autores.

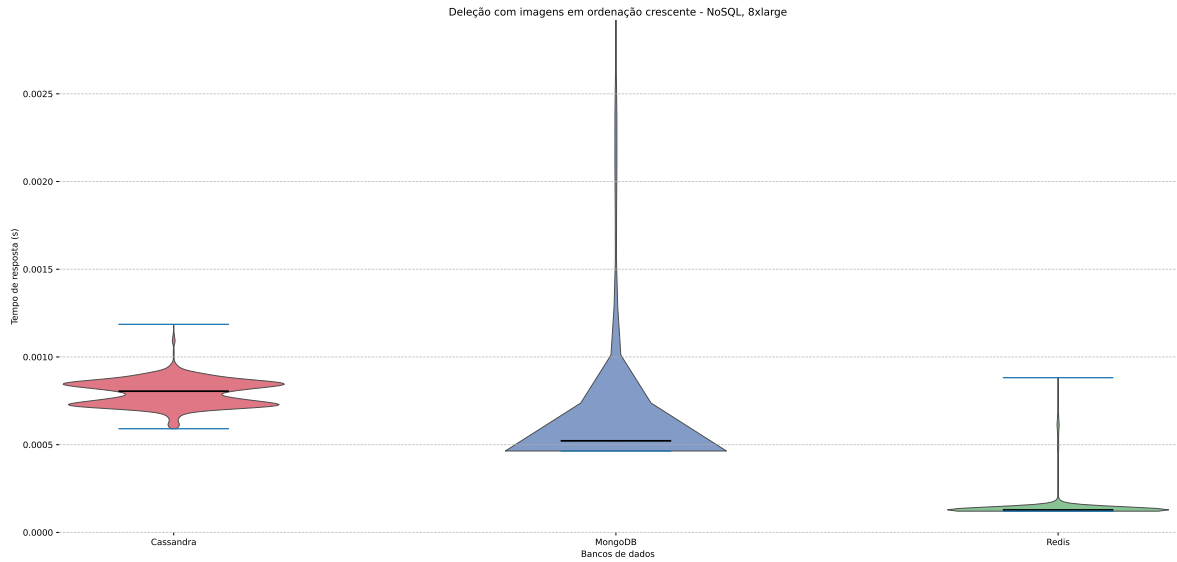

Figura A.87 – *Zoom* para inspeção das curvas do Cassandra, MongoDB e Redis da Figura A.86

Fonte: Produzido pelos autores.

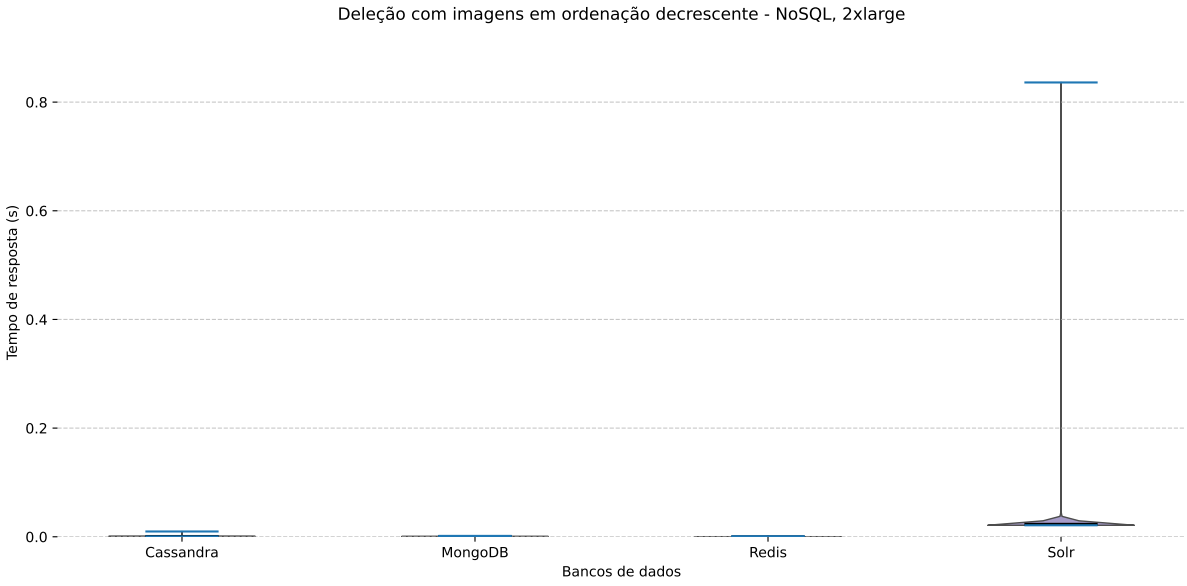

Figura A.88 – Gráfico dos tempos de resposta (deleção, decrescente, NoSQL, 2xlarge)

Fonte: Produzido pelos autores.

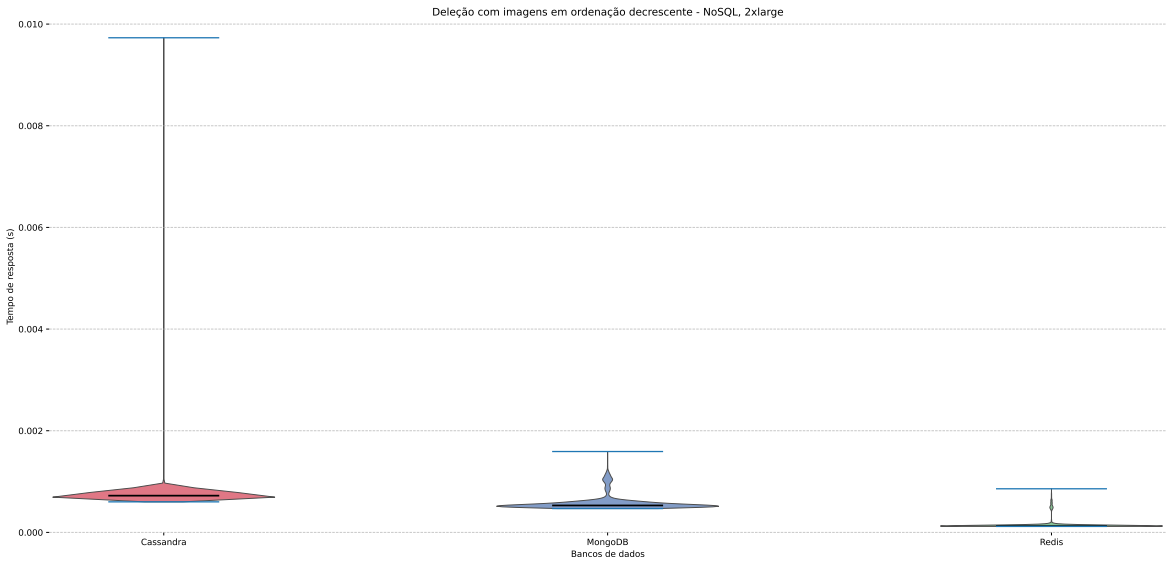

Figura A.89 – *Zoom* para inspeção das curvas do Cassandra, MongoDB e Redis da Figura A.88

Fonte: Produzido pelos autores.

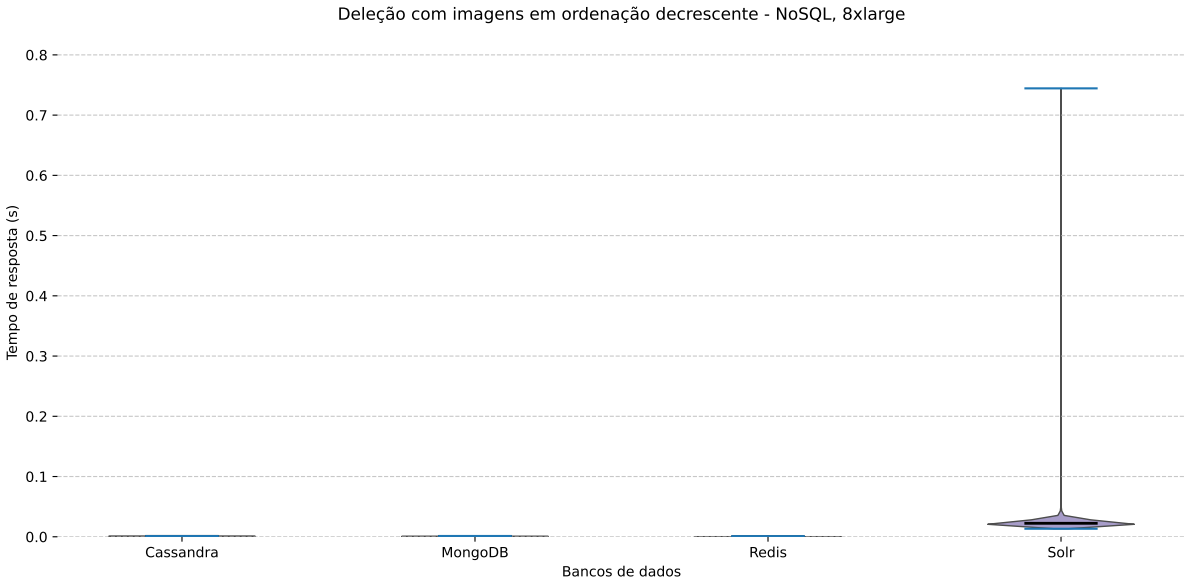

Figura A.90 – Gráfico dos tempos de resposta (deleção, decrescente, NoSQL, 8xlarge)

Fonte: Produzido pelos autores.

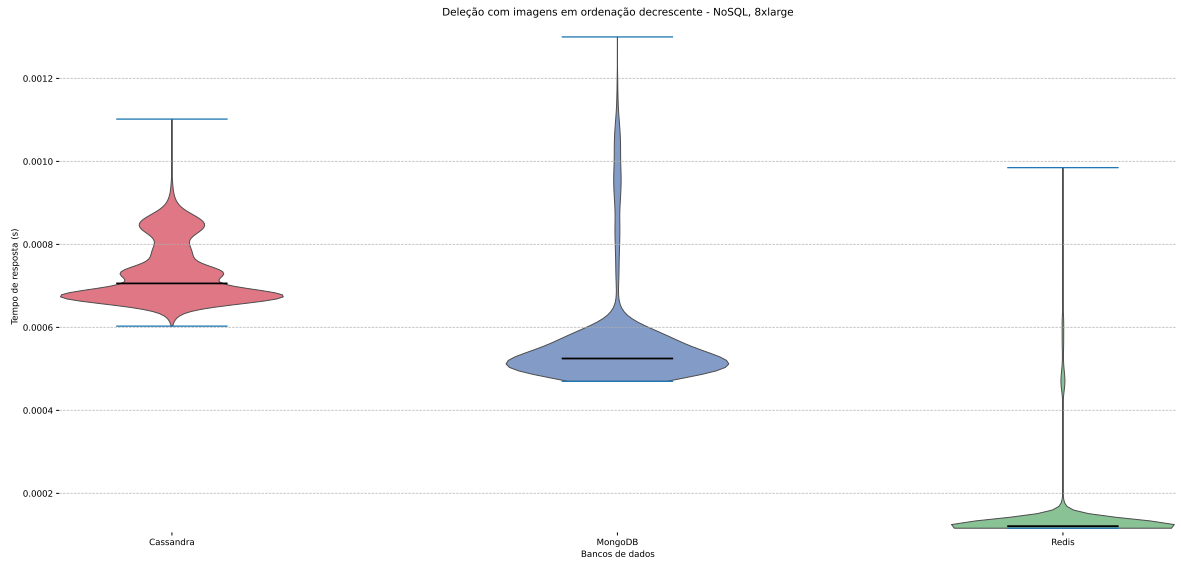

Figura A.91 – *Zoom* para inspeção das curvas do Cassandra, MongoDB e Redis da Figura A.90

Fonte: Produzido pelos autores.

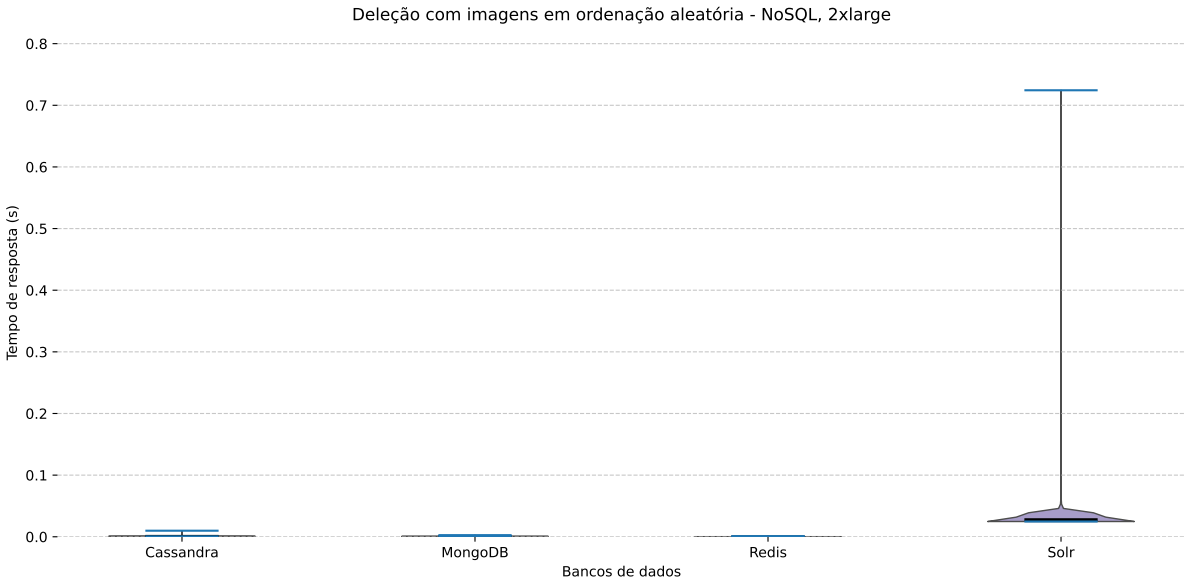

Figura A.92 – Gráfico dos tempos de resposta (deleção, aleatória, NoSQL, 2xlarge)

Fonte: Produzido pelos autores.

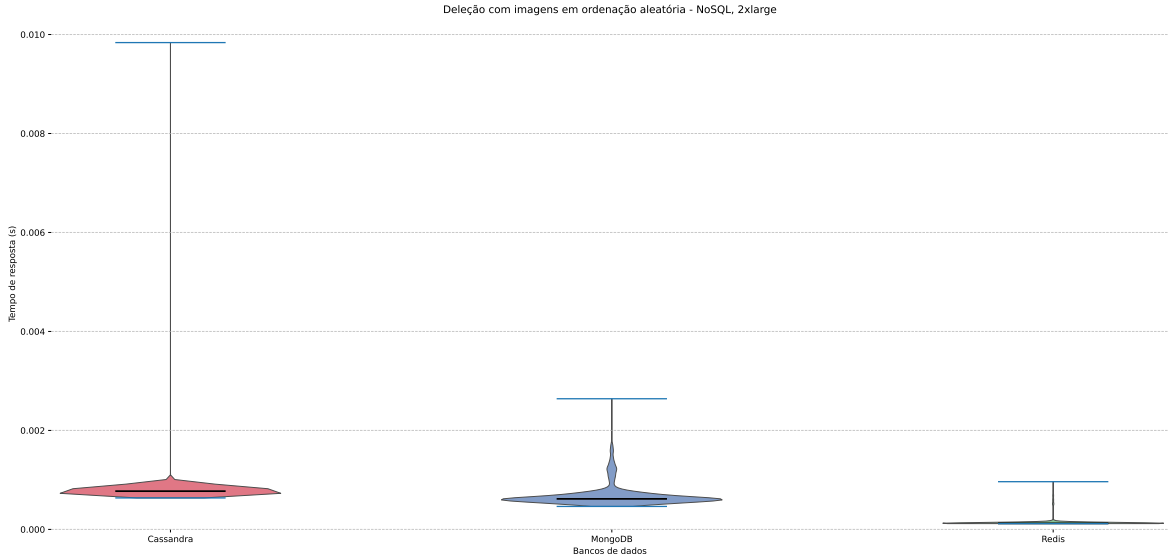

Figura A.93 – *Zoom* para inspeção das curvas do Cassandra, MongoDB e Redis da Figura A.92

Fonte: Produzido pelos autores.

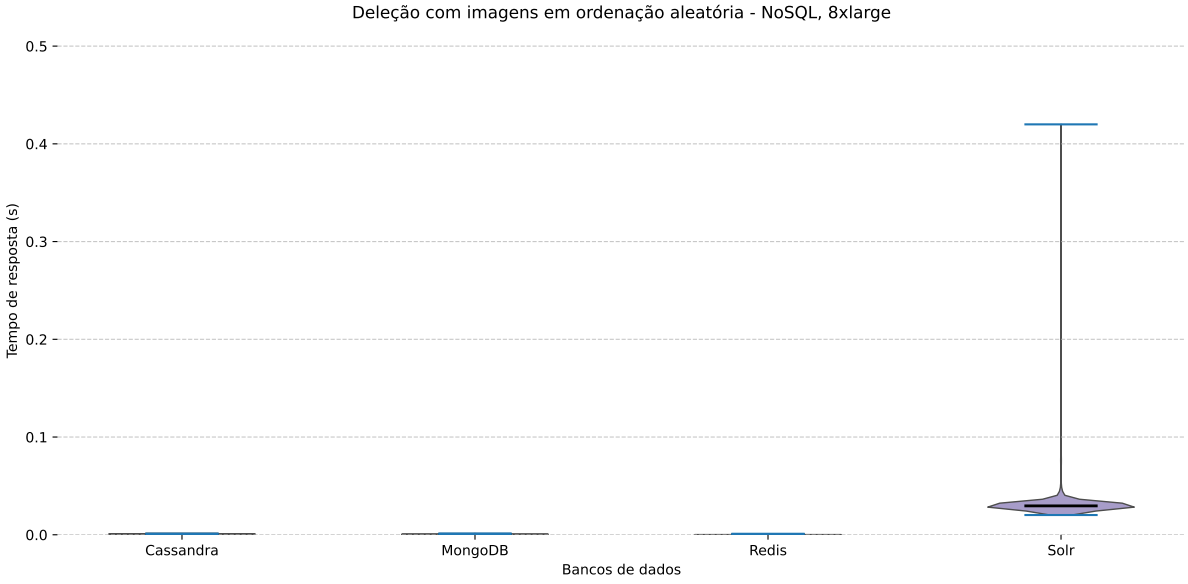

Figura A.94 – Gráfico dos tempos de resposta (deleção, aleatória, NoSQL, 8xlarge)

Fonte: Produzido pelos autores.

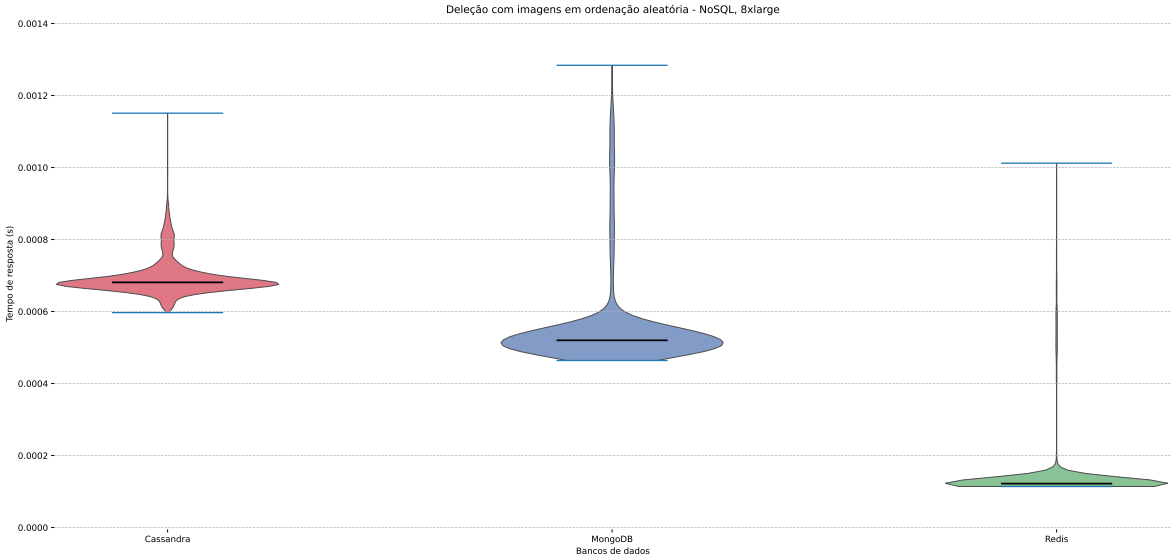

Figura A.95 – *Zoom* para inspeção das curvas do Cassandra, MongoDB e Redis da Figura A.94

Fonte: Produzido pelos autores.

Apêndice B – Gráficos do tipo setores para análise da distribuição dos resultados de cada ordenação

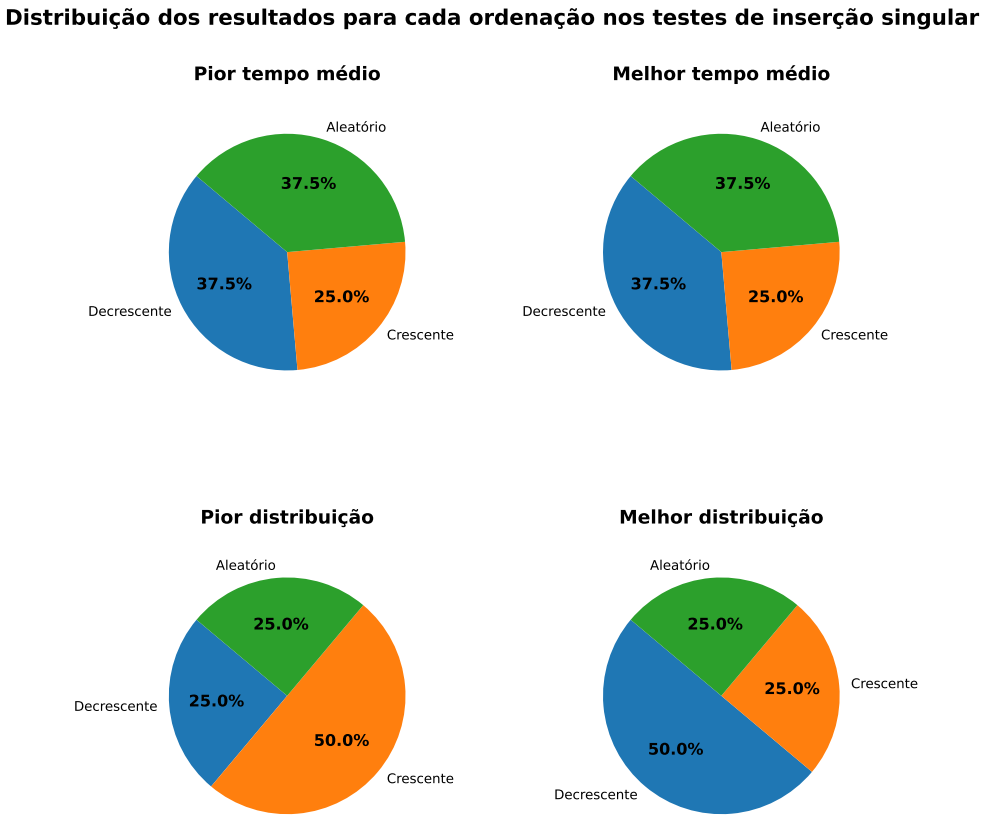

Figura B.96 – Gráfico da distribuição das ordenações por casos analisados para inserção singular

Fonte: Produzido pelos autores.

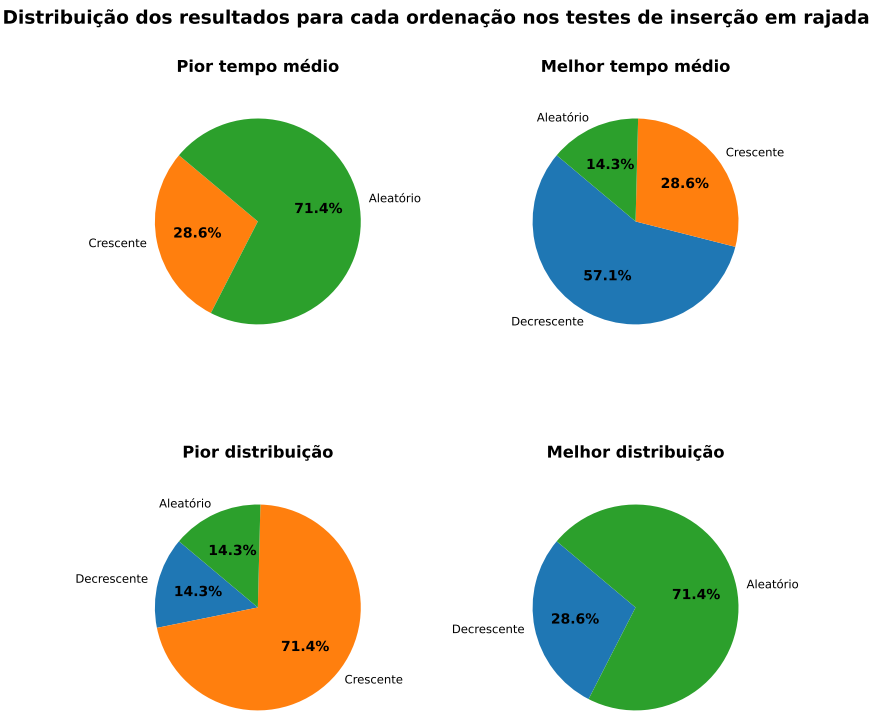

Figura B.97 – Gráfico da distribuição das ordenações por casos analisados para inserção em rajada

Fonte: Produzido pelos autores.

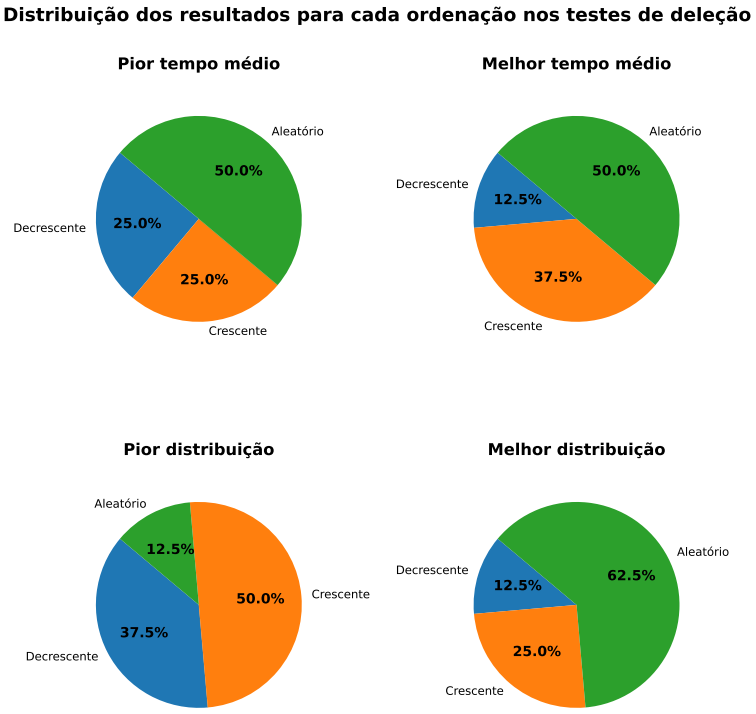

Figura B.98 – Gráfico da distribuição das ordenações por casos analisados para deleção

Fonte: Produzido pelos autores.