UNIVERSIDADE DE SÃO PAULO - USP
Instituto De Ciências Matemáticas e de Computação - ICMC

Profa. Elaine Parros Machado de Sousa

Trabalho de Arquitetura de SGBDs

SCC0243 - Arquitetura de Sistemas Gerenciadores de Base de Dados

Lívia Lelis 12543822
Lourenço de Salles Roselino 11796805
Samuel Figueiredo Veronez 12542626

São Carlos - SP
05/2025

Sumário

1 Indrodução . . . 3
2 Fundamentação Teórica . . . 4
2.1 Banco de Dados Distribuído . . . 4
2.1.1 Fragmentação Horizontal . . . 4
2.1.2 Fragmentação Vertical . . . 4
2.1.3 Fragmentação Mista . . . 4
2.1.4 Replicação . . . 4
2.1.5 Homogeneidade e Heterogeneidade . . . 4
2.1.6 Alocação de Dados . . . 5
2.1.7 Consulta em Banco de Dados Distribuído . . . 5
2.1.8 Plano de Consulta . . . 5
2.1.9 Transações . . . 5
2.2 Citus . . . 5
2.2.1 Nós Coordenadores e Nós Trabalhadores . . . 6
2.2.2 Tipos de Tabelas . . . 6
2.2.3 Transações . . . 6
3 Sistema Proposto . . . 7
3.1 Ferramentas Utilizadas . . . 7
4 Implementação . . . 8
5 Experimentos e Resultados . . . 9
5.1 Resultados do TPC-C . . . 9
6 Conclusão . . . 11
7 Trabalho Futuro . . . 11
8 Referências . . . 12

1 Indrodução

Conforme a evolução dos meios de comunicação, a capacidade de armazenamento, processamento e transmissão de dados aumentaram exponencialmente, e apenas um servidor tornou-se insuficiente para atender a demanda de usuários das grandes plataformas de serviços digitais de maneira efetiva e de alta disponibilidade.

Nesse contexto, para o projeto da disciplina SCC0243 - Arquitetura de Sistemas Gerenciadores de Base de Dados, decidimos comparar a performance de um banco de dados distribuído com um banco de dados centralizado, utilizando o PostgreSQL e o Citus, uma extensão do PostgreSQL para bancos de dados distribuídos.

O objetivo do projeto é implementar e avaliar as diferenças de performance e overheads de uma base de dados relacional distribuída do Citus, em comparação ao PostgreSQL centralizado. Para isso, utilizaremos um benchmark padronizado pelo Concelho de Desempenho de Processamento de Transações(TPC, em inglês).

Todos os códigos e instruções para a execução do projeto estão disponíveis no repositório do projeto, disponível em <https://github.com/VH-Evil-Inc/asgbd>.

2 Fundamentação Teórica

2.1 Banco de Dados Distribuído

Um banco de dados distribuído é um banco de dados cujos dados estão armazenados em diferentes locais, mas que se intercomunicam de forma a parecerem um único banco de dados para o usuário.

A distribuição dos dados entre várias máquinas possibilita melhorar a escalabilidade, a disponibilidade e a tolerância a falhas do sistema. Entretanto, essa distribuição implica em custos adicionais de comunicação e sincronização entre os nós do sistema.

No contexto da distribuição dos dados em bancos de dados distribuídos, tanto as tuplas quanto os atributos que as compõem podem ser fragmentados entre os nós do sistema, e cópias desses dados podem ser replicadas em mais de um nó.

2.1.1 Fragmentação Horizontal

A fragmentação horizontal consiste na distribuição das tuplas inteiras de uma tabela entre diferentes máquinas do sistema. Essa fragmentação é realizada de modo que cada fragmento contenha um subconjunto das tuplas da tabela original, mantendo todas as colunas.

2.1.2 Fragmentação Vertical

A fragmentação vertical consiste na distribuição dos atributos de uma tabela entre diferentes máquinas do sistema, mantendo em comum a chave primária da tabela original. Dessa forma, cada fragmento armazena um subconjunto dos atributos para cada tupla da tabela original.

2.1.3 Fragmentação Mista

A fragmentação mista combina as estratégias de fragmentação horizontal e vertical. Nesse caso, cada fragmento armazena um subconjunto dos atributos de uma tabela para um subconjunto das tuplas.

2.1.4 Replicação

No que diz respeito à redundância dos dados entre os nós do sistema, a replicação consiste na cópia de um fragmento de dados em mais de um nó. Essa cópia pode ser feita de forma parcial ou completa, e uma maior redundância dos dados implica em maior disponibilidade e tolerância a falhas do sistema, mas aumenta os custos de armazenamento e sincronização.

2.1.5 Homogeneidade e Heterogeneidade

Os bancos de dados distribuídos também podem ser compostos por nós de mesma natureza, no que diz respeito ao SGBD utilizado, modelos de dados, protocolos de comunicação, fragmentação e replicação dos dados.

Um sistema cujos nós seguem a mesma arquitetura geral é classificado como homogêneo, e apresenta maior desempenho geral, simplicidade de implementação e escalabilidade.

Por outro lado, um sistema heterogêneo é composto por nós de diferentes arquiteturas, o que aumenta a complexidade de implementação e manutenção, além de exigir um esforço maior de intercomunicação entre os nós para garantir a transparência ao usuário.

2.1.6 Alocação de Dados

A alocação de dados em um banco de dados distribuído é o processo pelo qual se decide onde os dados serão armazenados, levando em consideração a configuração dos nós do sistema e as métricas de desempenho desejadas. O processo de alocação de dados pode ser feito de forma centralizada ou descentralizada.

2.1.7 Consulta em Banco de Dados Distribuído

Devido à variedade de configurações e aos fragmentos de dados contidos nos nós do sistema, o processo de consulta se torna mais complexo em um banco de dados distribuído.

2.1.8 Plano de Consulta

O plano de consulta em bancos de dados distribuídos é a sequência de operações que define como uma consulta será executada sobre os dados fragmentados e possivelmente replicados entre diferentes nós. O otimizador do SGBD distribui as operações entre os nós, buscando minimizar o custo de comunicação e o tempo de resposta. O plano pode envolver o envio de subconsultas para diferentes nós, a coleta e combinação dos resultados parciais, e a aplicação de operações finais no nó coordenador. A eficiência do plano de consulta é fundamental para o desempenho do sistema distribuído.

2.1.9 Transações

Transações em bases de dados distribuídas envolvem múltiplos nós ou locais onde os dados estão armazenados, mas precisam ser executadas de forma coordenada para garantir as propriedades de Atomicidade, Consistência, Isolamento e Durabilidade (ACID). Mesmo que uma transação envolva atualizações em diferentes servidores, ela deve ser concluída integralmente em todos eles ou ser completamente desfeita em caso de falha, mantendo o sistema em um estado consistente.

Para garantir essas propriedades, sistemas distribuídos utilizam protocolos especiais, como o protocolo de commit em duas fases (Two-Phase Commit – 2PC), que coordenam a aprovação e execução das operações entre todos os nós participantes. Esses mecanismos são essenciais para evitar inconsistências e garantir a confiabilidade das transações.

2.2 Citus

Para o desenvolvimento do projeto, foi utilizado o Citus, uma extensão do PostgreSQL para bancos de dados distribuídos. O Citus apresenta suporte à fragmentação horizontal, fragmentação por esquema e replicação de dados. O Citus suporta exclusivamente o SGBD PostgreSQL.

2.2.1 Nós Coordenadores e Nós Trabalhadores

O Citus atribui aos seus nós dois papéis distintos: worker e coordinator.

Os worker nodes são responsáveis por armazenar os dados e processar as consultas, compondo a maior parte dos nós do sistema.

Cada cluster de nós trabalhadores é gerenciado por um nó coordenador, cujo papel é intermediar as consultas da aplicação entre um ou múltiplos nós que contêm os dados desejados, acumular os resultados e retornar o resultado final para a aplicação. O nó coordenador também mantém a alocação, o controle da consistência dos dados e da integridade dos nós trabalhadores.

2.2.2 Tipos de Tabelas

O Citus oferece os seguintes tipos de tabelas:

- Tabelas distribuídas (distributed tables): São tabelas cujos dados são particionados (sharded) e distribuídos entre vários nós trabalhadores. Cada linha é atribuída a um shard com base em uma coluna de distribuição escolhida, permitindo consultas e operações paralelas.
- Tabelas de referência (reference tables): São tabelas replicadas em todos os nós trabalhadores, utilizadas para armazenar dados pequenos e frequentemente acessados.
- Tabelas locais (local tables): São tabelas que existem apenas no nó coordenador e não são distribuídas nem replicadas. Funcionam como tabelas normais do PostgreSQL e são úteis para dados administrativos ou de controle.

2.2.3 Transações

O Citus implementa o protocolo de commit em duas fases (2PC) para garantir as propriedades ACID e utiliza mecanismos adicionais para lidar com falhas, como o uso de metadados distribuídos para recuperação de transações incompletas.

3 Sistema Proposto

O sistema proposto foi desenvolvido para comparar o desempenho de uma instância única do PostgreSQL com um ambiente em cluster utilizando o Citus.

O benchmark utilizado será o TPC-C, desenvolvidos pelo TPC (Transaction Processing Council), amplamente reconhecido como um padrão do mercado para avaliar desempenho e escalabilidade de sistemas de banco de dados. A execução do benchmark será intermediada pelo framework HammerDB. Utilizaremos o grafana e o prometheus para monitorar o sistema durante os testes.

Para garantir fácil reprodutibilidade e configuração, ambos os ambientes serão configurados utilizando o Docker Compose.

3.1 Ferramentas Utilizadas

- Docker Compose: Ferramenta de orquestração de containers empregada para configurar e iniciar facilmente os ambientes de banco de dados, tanto para a instância única quanto para o cluster.
- PostgreSQL: Utilizado como sistema gerenciador de banco de dados relacional base, tanto no modo standalone quanto no cluster.
- Citus: Extensão do PostgreSQL que permite a distribuição dos dados e consultas entre múltiplos nós, viabilizando o escalonamento horizontal do banco de dados.
- Grafana e Prometheus: Utilizados para monitoramento e análise de métricas do sistema. O Prometheus coleta e armazena as métricas do PostgreSQL, enquanto o Grafana fornece dashboards interativos, para visualização em tempo real do comportamento do sistema.
- HammerDB: Framework de benchmarking utilizado para executar testes de carga nos bancos de dados. Foi empregado o benchmark TPC-C, reconhecido como padrão do mercado para avaliação de desempenho e escalabilidade de sistemas de banco de dados.
- TPC-C: O TPC-C é um benchmark amplamente utilizado para medir o desempenho de sistemas de banco de dados em ambientes de processamento de transações online (OLTP). Ele simula o funcionamento de uma empresa atacadista, incluindo operações como criação de pedidos, pagamentos, entregas e controle de estoque, por meio de cinco tipos de transações típicas do dia a dia empresarial. O teste avalia a capacidade do sistema de processar grandes volumes de transações simultâneas, refletindo cenários reais de negócios, e utiliza como principal métrica o número de transações por minuto (tpmC), permitindo comparar a performance e escalabilidade de diferentes soluções de banco de dados.

  O esquema relacional do TPC-C é o seguinte:

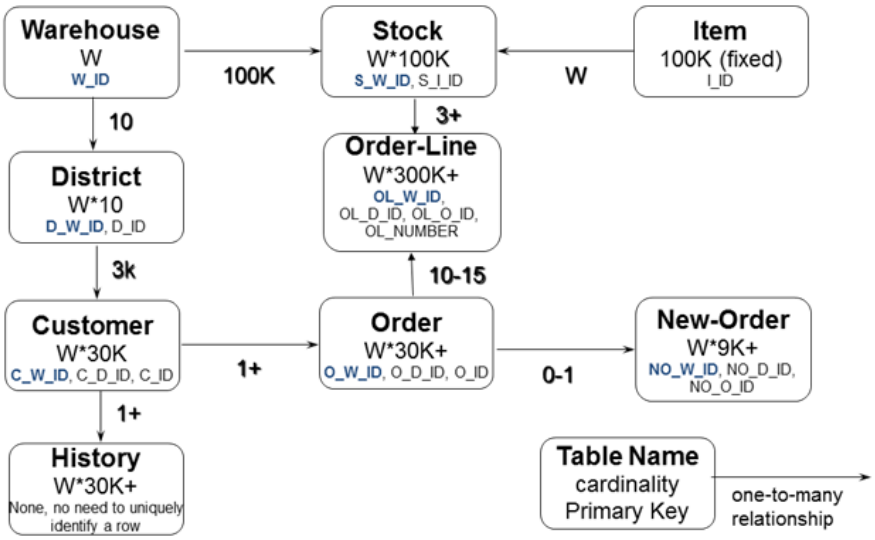

Figura 1 – obtido em: https://www.hammerdb.com/docs3.3/ch03s05.html

4 Implementação

O sistema foi implementado utilizando o Docker Compose, com as ferramentas explicadas na sessão anterior, e está disponível no repositório do projeto.

Os scripts de inicialização dos ambientes e execução dos benchmarks foram automatizados, facilitando a replicação dos experimentos e a coleta dos resultados.

Os comandos para reproduzir o sistema e executar o benchmark estão disponíveis no readme do repositório.

5 Experimentos e Resultados

Para comparar o desempenho entre o PostgreSQL e o Citus, executamos os benchmarks TPC-C em ambos os sistemas. No caso do citus, utilizamos uma configuração de 3 nós trabalhadores e 1 nó coordenador. Além disso, para simular um ambiente de cloud, cada instância foi limitada a 8GB de RAM e 2 CPUs, além da introdução de latência de rede aleatória (150us ± 200us).

A carga é executada e em seguida o benchmark executa durante um período 5 minutos, com 64 conexões simultâneas, e parâmetro do benchmark warehouses = 40.

Os resultados foram coletados e analisados utilizando o Grafana e o Prometheus, e serão discutidos nas seções a seguir.

Vale ressaltar que no caso dos gráficos do Citus, os resultados foram coletados do nó coordenador, que é o responsável por coordenar as consultas e distribuir as tarefas entre os nós trabalhadores.

Os gráficos a seguir mostram o desempenho do sistema durante a execução do benchmark:

5.1 Resultados do TPC-C

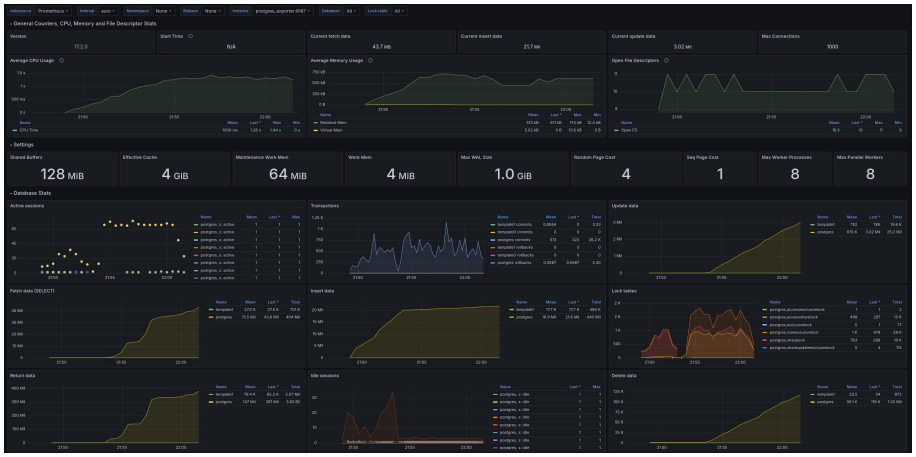

Figura 2 – Resultados do TPC-C para o Citus

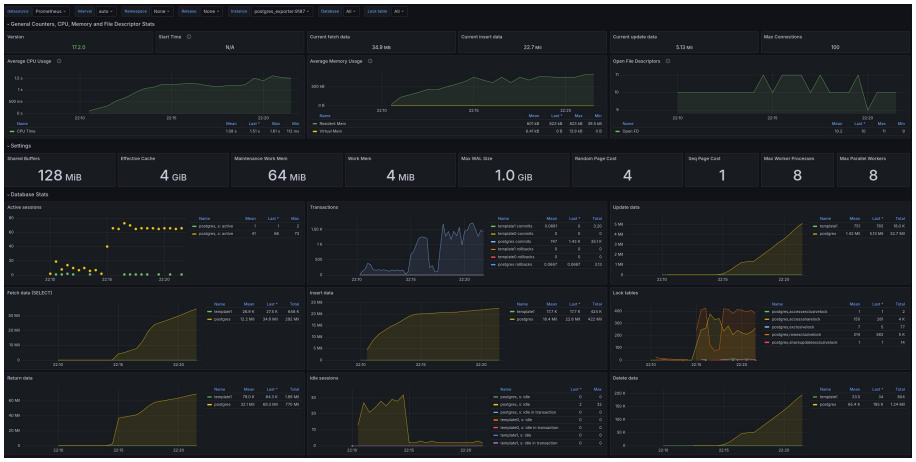

Figura 3 – Resultados do TPC-C para o PostgreSQL

Como resultados, obtemos que O PostgreSQL atingiu 31405 operações novas por minuto e 72424 transações por minuto, enquanto o Citus atingiu 15304 operações novas por minuto e 35762 transações por minuto. Além disso, o tempo das operações está registrado nas tabelas a seguir:

PROC	Banco	MIN (ms)	AVG (ms)	MAX (ms)	TOTAL (ms)
PAYMENT	*Postgres*	0.833	52.601	1257.365	34106475.135
	Citus	0.990	101.086	1767.711	36185664.966
NEWORD	*Postgres*	1.640	48.569	1255.893	31475935.439
	Citus	1.774	107.871	1737.546	38724729.750
SLEV	*Postgres*	0.440	153.889	21302.445	10003419.917
	Citus	0.612	63.438	5600.833	2288143.726
DELIVERY	*Postgres*	1.278	54.623	1417.772	3534810.624
	Citus	2.286	115.989	1505.085	4142561.285
OSTAT	*Postgres*	0.104	2.725	455.587	173186.360
	Citus	0.269	7.474	609.841	262082.364

PROC	Banco	P99 (ms)	P95 (ms)	P50 (ms)
PAYMENT	*Postgres*	413.035	167.278	27.633
	Citus	611.674	352.694	72.123
NEWORD	*Postgres*	413.374	160.868	23.915
	Citus	607.420	362.191	83.081
SLEV	*Postgres*	4235.829	73.669	6.215
	Citus	730.162	224.685	8.710
DELIVERY	*Postgres*	441.496	191.119	25.470
	Citus	596.206	375.478	90.453
OSTAT	*Postgres*	47.199	3.844	1.292
	Citus	80.547	60.180	2.439

Nesse contexto, os resultados apresentados evidenciam o overhead do Citus, especialmente em cenários de alta concorrência simulados pelo benchmark TPC-C.

O PostgreSQL standalone alcançou 31405 operações novas por minuto e 72424 transações por minuto, enquanto o clusterizado atingiu 15304 operações novas por minuto e 35762 transações por minuto. Isso indica que o PostgreSQL foi capaz de processar aproximadamente o dobro de operações e transações por minuto em relação ao Citus, evidenciando que, nesse cenário, o overhead do Citus superou os benefícios da distribuição de dados e consultas. Além disso, o tempo médio registrado pelas operações do Citus foi significativamente maior em comparação ao PostgreSQL.

Considerando que ambos os testes foram realizados em um ambiente conteinerizado, é possível que algumas limitações de hardware, como a capacidade de leitura e escrita do disco possam ter influenciado o gargalo observado no Citus.

6 Conclusão

Os resultados demonstram que, para o cenário avaliado, o PostgreSQL standalone apresentou desempenho superior ao Citus, tanto em throughput quanto em latência das operações. Embora o Citus ofereça vantagens em escalabilidade e distribuição de dados, acreditamos que o fato de ter sido executado em uma única máquina e com apenas um disco rígido dividido entre todos os nós possa ter influenciado negativamente o desempenho geral do sistema. Devido ao possível viés introduzido pelo ambiente, conclusões não são definitivas e devem ser interpretadas com cautela.

7 Trabalho Futuro

Para o futuro, sugerimos a realização de testes adicionais em um ambiente em cloud, onde a infraestrutura possa ser escalada horizontalmente, simulando um cenário mais próximo de um ambiente de produção.

Além disso, a execução de outros benchmarks, poderia fornecer uma visão mais abrangente do desempenho do sistema.

8 Referências

- PostgreSQL - <https://www.postgresql.org/>
- Citus Documentation - <https://www.docs.citusdata.com/>
- TPC-C Benchmark - <https://www.tpc.org/TPC_Documents_Current_Versions/pdf/tpc-c_v5.11.0.pdf>
- HammerDB - <https://www.hammerdb.com/>
- Grafana - <https://grafana.com/>
- Prometheus - <https://prometheus.io/>