Comparativo entre bases de dados distribuídas SQL e NoSQL

Análises e resultados de sistemas e estratégias

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

SCC0243 - Arquitetura de Sistemas Gerenciadores de Base de Dados

São Carlos - SP | 1º Semestre 2025 ICMC-USP

Introdução & Motivação

Tendo em vista a demanda crescente por sistemas eficientes e robustos para dados este trabalho foi desenvolvido para avaliar a performance e overhead entre diferentes sistemas gerenciadores de base de dados, distribuídos e centralizados.

Este trabalho não visa fazer uma comparação direta entre a performance individual de cada SGBD e sim estudar e entender suas técnicas de distribuição e comparar seus escalonamentos horizontais.

Conceitos para Bases de Dados Distribuídas

- Fragmentação
  - Horizontal
  - Vertical
  - Mista
- Replicação
- Homogenidade e Heterogenidade
- Alocação de dados

PostgreSQL & Citus

Arquitetura usada na parte de psql

Citus

Citus é uma extensão do PostgreSQL para bancos de dados distribuídos. O Citus apresenta suporte à fragmentação horizontal, fragmentação por esquema e replicação de dados.

Modelos de fragmentação

Fragmentação Horizontal

Um único esquema é utilizado entre todos os nós, tendo a fragmentação feita por tuplas que tem seu nó de residência determinado por uma coluna de distribuição.

Fragmentação por Esquema

Permite diferentes nós usarem diferentes esquemas, ainda permite tabelas compartilhadas e outras funcionalidades do Citus em esquemas não fragmentados.

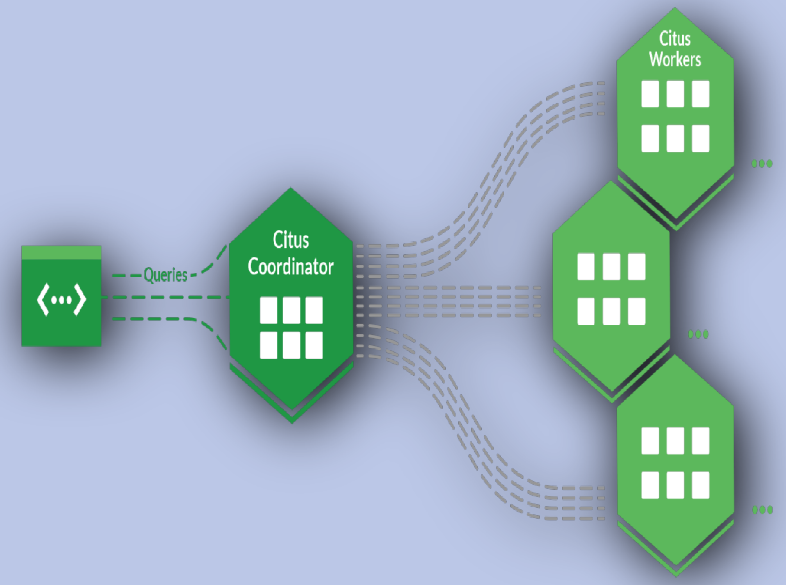


Figura: Arquitetura coordinator-worker do Citus. Obtido em: https://github.com/citusdata/citus#Architecture

- Dois tipos de nós: worker e coordinator
- Worker armazenam os dados das tabelas distribuidas e processam as consultas
- Cada coordinator node gerencia um cluster de worker nodes
  - Faz o intermédio das consultas da aplicação entre um ou múltiplos nós, acumulando e retornando os resultados para a aplicação
  - Também também mantém a alocação, o controle da consistência dos dados e da integridade dos nós trabalhadores.

Tipos de Tabelas

Tabelas distribuídas

Tabelas com os dados particionados e distribuídos entre vários nós trabalhadores, permitindo consultas e operações paralelas.

Tabelas de referência

São tabelas replicadas em todos os nós trabalhadores, utilizadas para armazenar dados pequenos e frequentemente acessados.

Tabelas locais

São tabelas que existem apenas no nó coordenador e não são distribuídas nem replicadas.

TCP-C ! :)

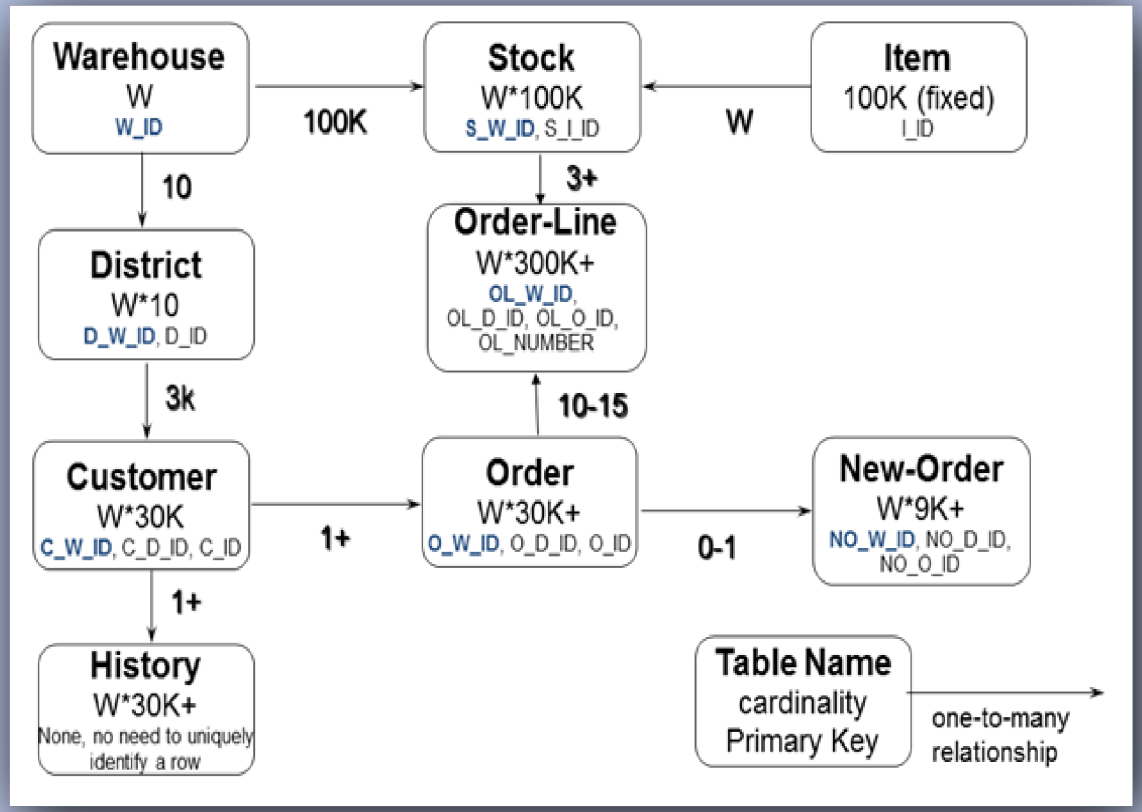


Figura: Esquema relacionado do TPC-C. Obtido em: https://www.hammerdb.com/docs3.3/ch03s05.html

Estratégia de Distribuição

```
-- Distribution Configuration
SELECT create_reference_table('warehouse');
SELECT create_reference_table('district');
SELECT create_reference_table('item');

SELECT create_distributed_table('customer', 'c_w_id', colocate_with => 'none');
SELECT create_distributed_table('stock', 's_w_id', colocate_with => 'customer');
SELECT create_distributed_table('orders', 'o_w_id', colocate_with => 'customer');
SELECT create_distributed_table('new_order', 'no_w_id', colocate_with => 'customer');
SELECT create_distributed_table('order_line', 'ol_w_id', colocate_with => 'customer');
SELECT create_distributed_table('history', 'h_w_id', colocate_with => 'customer');
```

Resultados

Cassandra 👁

Arquitetura usada na parte de nosql

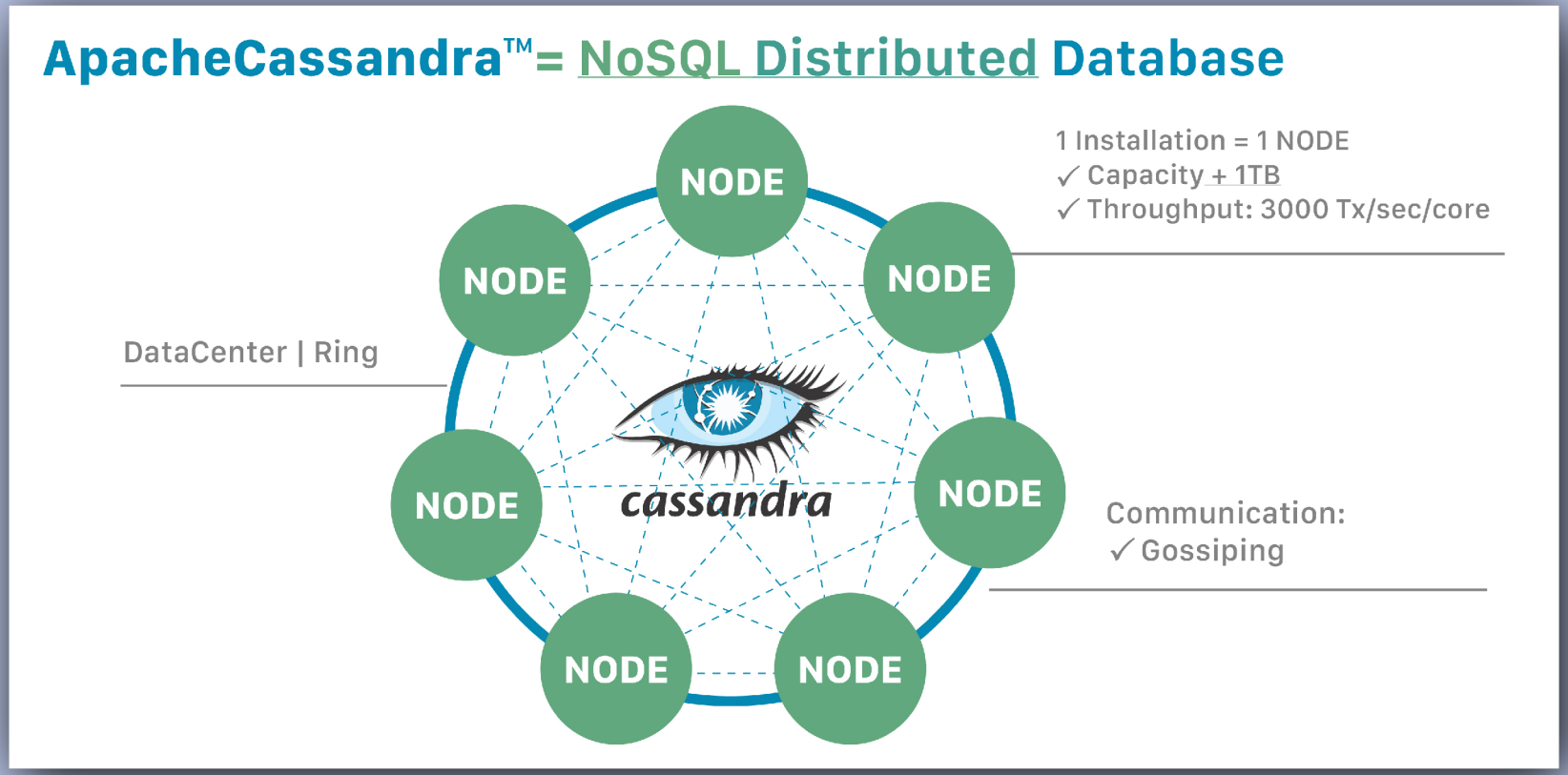
Talvez falar sobre Wide Column ?

Cassandra

Apache Cassandra é um banco de dados **open-source** NoSQL distribuído, sendo classificado como um **Wide-Column Database**.

Cassandra

Cassandra utiliza uma arquitetura **masterless** com **clusters** organizados em forma de anel, todos os nós do Cassandra podem responder a aplicação e comunicar com todos os outros nós dentro de um anel.



Estrutura de Dados e Particionamento

Os dados no cassandra são organizados em keyspaces que agrupam tabelas relacionadas. Cada tabela é composta por linhas e colunas

Cada linha é identificada por uma chave primária composta por um partition key e, opcionalmente, colunas de ordenação (clustering columns)

A distribuição dos dados é feita de forma que cada nó do cluster seja responsável por um intervalo do espaço de tokens.

Replicação e Tolerância a Falhas

Cassandra implementa replicação configurável por **keyspace**, permitindo definir o fator de replicação e a estratégia de replicação mais adequada ao ambiente.

Consistência

O Cassandra possui **tunable consistency**, permitindo o usuário definir por operação quantos nós precisam confirmar uma leitura ou escrita para que ela seja bem-sucedida.

Isso permite ajustar entre priorizar consistência forte ou disponibilidade, conforme a necessidade da aplicação.

Por padrão o Cassandra opera como um sistema **AP** (alta disponibilidade e tolerância a partições), mas pode ser configurado para comportar-se como **CP** (consistência e tolerância a partições) em cenários específicos.

Yahoo! Cloud Serving Benchmarking (YCSB)

O YCSB é um benchmark amplamente utilizado para avaliar o desempenho de sistemas de banco de dados NoSQL.

Para este projeto, utilizaremos o YCSB para executar uma série de testes em cada configuração do Cassandra, com foco em medir o throughput e a latência das operações.

O YCSB utiliza um modelo de dados simples baseado em chave-valor. O formato padrão do banco de dados é uma tabela chamada geralmente de usertable, que possui: 1 chave primária `YCSB_KEY` e um conjunto de dados `FIELD0`, `FIELD1`, ..., `FIELD9` que por padrão são tipo String.

Resultados

Configuração	Tempo de Carregamento (ms)	Tempo de Execução (ms)
Nó único	386 275	625 923
3 nós sem replicação	224 126	225 164
3 nós com replicação	477 892	552 229

Conclusão

Bibliografia

- uau documentação

Perguntas!