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, configurações e comparar os beneficios de seus escalonamentos horizontais.

Conceitos para Bases de Dados Distribuídas

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

Citus

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, não permite junções entre esquemas distribuidos nem paralelização de queries mas ainda permite tabelas de referência em esquemas não fragmentados.

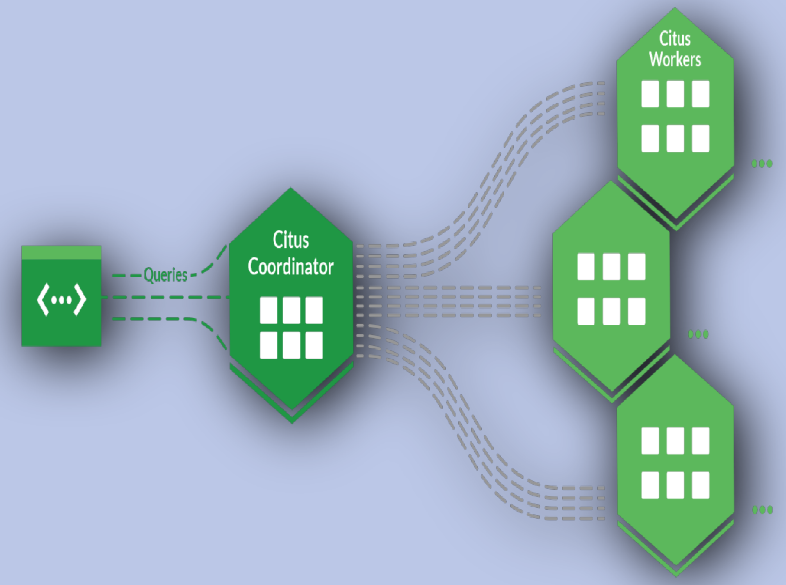


Figura: Arquitetura coordinator-worker do Citus. Obtido em: https://github.com/citusdata/citus#Architecture

- 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.

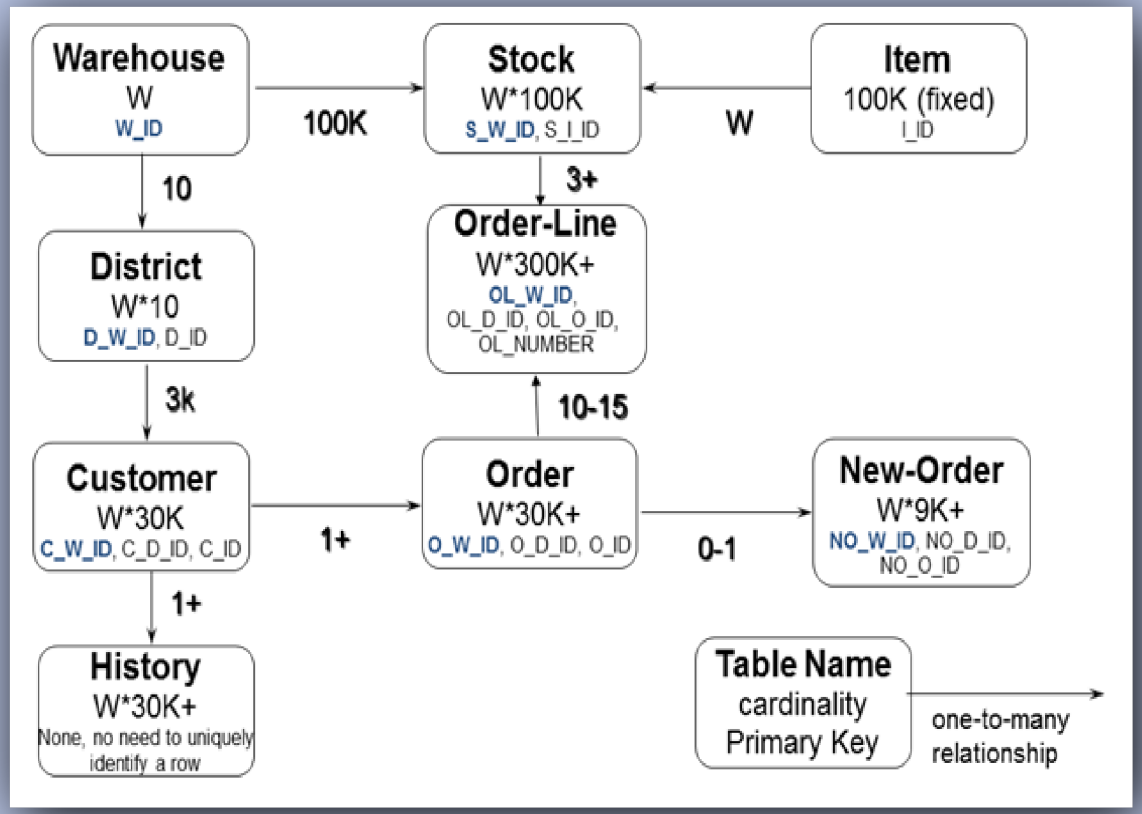


Figura: Esquema relacionado do TPC-C. Obtido em: https://www.hammerdb.com/docs3.3/ch03s05.html

TCP-C

- Benchmark padronizado pela Transaction Processing Performance Council.
- Simula um sistema de empresa de vendas por atacado e transações típicas.
  - Novos Pedidos
  - Pagamentos
  - Processamento de Entregas
  - Consulta de Status de Pedidos
  - Consulta de Níveis de Estoque

Estratégia de Distribuição do Citus

```
-- Distribution Configuration
SELECT create_distributed_table('customer', 'c_w_id')
SELECT create_distributed_table('district', 'd_w_id')
SELECT create_distributed_table('history', 'h_w_id')
SELECT create_distributed_table('warehouse', 'w_id')
SELECT create_distributed_table('stock', 's_w_id')
SELECT create_distributed_table('new_order', 'no_w_id')
SELECT create_distributed_table('orders', 'o_w_id')
SELECT create_distributed_table('order_line', 'ol_w_id')

SELECT create_reference_table('item')
```

Ambiente

- Single Node x Multi Node (1 coord. ; 3 workers)
- Inicial: 4 CPUs e 8GB de RAM, restringidos via Docker no mesmo host
- Cloud: 4 vCPUs e 8GB de RAM, droplets de recurso compartilhado na DO

Resultados do Benchmark (Local)

- Single-Node : 31.405 operações novas e 62.424 transações por minuto
- Multi-Node : 15.304 operações novas e 35.762 transações por minuto

PROC	Replicas	MIN (ms)	AVG (ms)	MAX (ms)	P99 (ms)	P95 (ms)	P50 (ms)
PAYMENT	Single	0.833	52.601	1257.365	413.035	167.278	27.633
	Multi	0.990	101.086	1767.711	611.674	352.694	72.123
NEWORD	Single	1.640	48.569	1255.893	413.374	160.868	23.915
	Multi	1.774	107.871	1737.546	607.420	362.191	83.081
SLEV	Single	0.440	153.889	21302.445	4235.829	73.669	6.215
	Multi	0.612	63.438	5600.833	730.162	224.685	8.710
DELIVERY	Single	1.278	54.623	1417.772	441.496	191.119	25.470
	Multi	2.286	115.989	1505.085	596.206	375.478	90.453
OSTAT	Single	0.104	2.725	455.587	47.199	3.844	1.292
	Multi	0.269	7.474	609.841	80.547	60.180	2.439

Resultados do Benchmark (Cloud)

- Single-Node : 51.498 operações novas e 118.229 transações por minuto
- Multi-Node : 67.108 operações novas e 154.437 transações por minuto

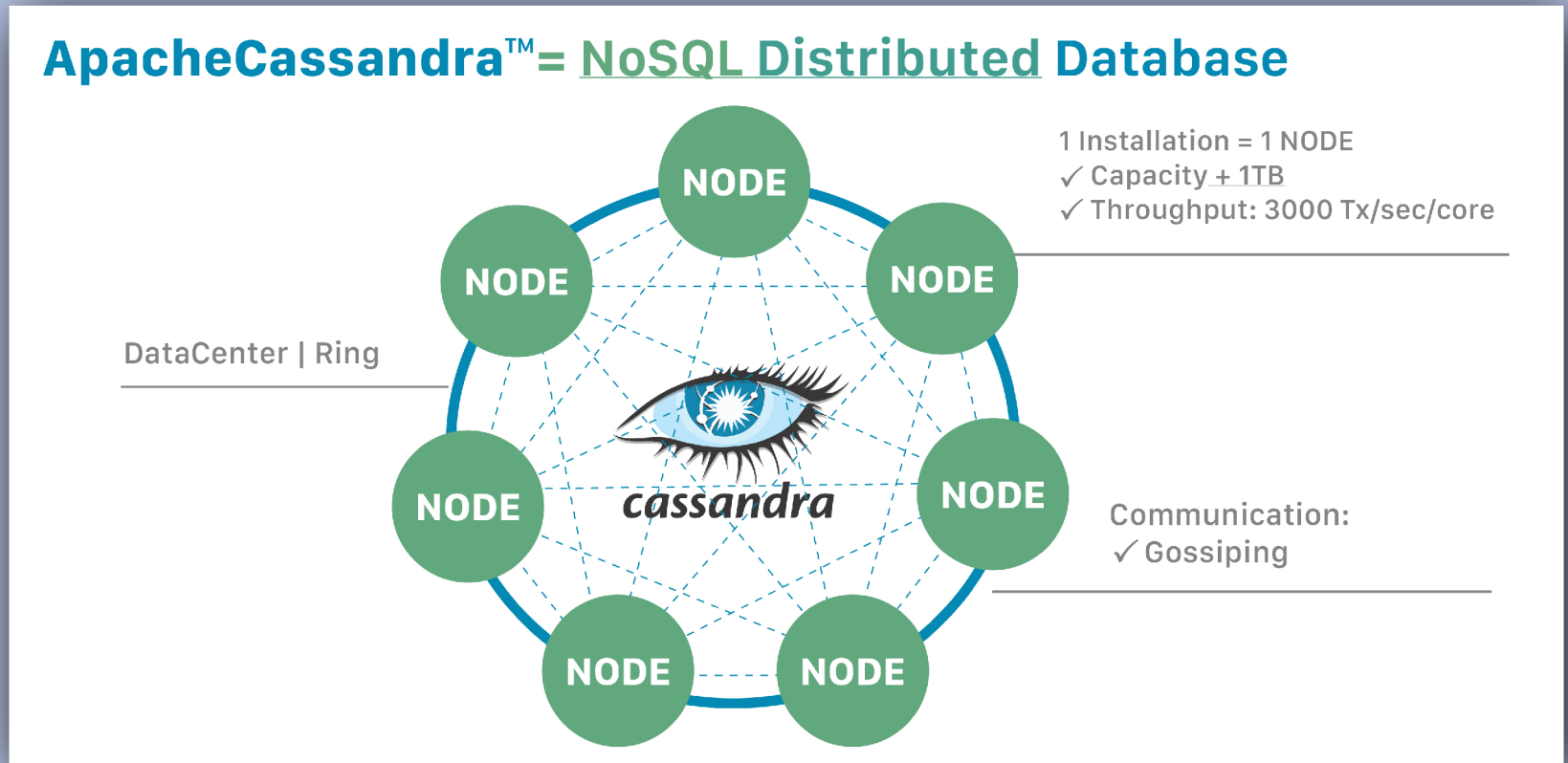
PROC	Replicas	MIN (ms)	AVG (ms)	MAX (ms)	P99 (ms)	P95 (ms)	P50 (ms)
NEWORD	Single	0.857	47.773	497.174	191.415	127.932	38.338
	Multi	2.333	26.435	582.658	211.060	113.347	9.682
PAYMENT	Single	0.400	15.545	422.251	112.142	50.166	9.886
	Multi	1.028	22.988	767.196	238.873	138.448	4.133
DELIVERY	Single	0.990	74.445	667.467	268.406	184.238	62.080
	Multi	2.539	26.001	388.739	144.493	89.586	13.082
SLEV	Single	0.926	22.806	14.701.909	130.593	54.970	7.961
	Multi	1.629	31.134	12.533.507	672.486	56.177	4.754
OSTAT	Single	0.400	10.674	364.984	63.122	34.078	6.585
	Multi	0.990	8.749	244.090	62.440	33.984	3.864

Cassandra 👁

Cassandra

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

- Arquitetura **masterless** com **clusters** organizados em forma de anel



Wide Column

- Organização em linhas e colunas, com formatos que podem variar para uma mesma tabela.
- Chave Primária definida como chave de partição e, opcionalmente, chave de clustering.

Estrutura de Dados e Particionamento

- Organização em **keyspaces**, distribuição por intervalod do espaço de tonkens
- Cada linha é identificada por uma chave primária composta por um partition key e, opcionalmente, colunas de ordenação

Replicação, Tolerância a Falhas e Consistência

- Configurável por **keyspace**, permitindo definir o fator e a estratégia de replicação.
- **Tunable consistency**, permitindo o usuário definir por operação quantos nós precisam confirmar uma leitura ou escrita para que ela seja bem-sucedida.
- Por padrão opera como sistema **AP** (alta disponibilidade e tolerância a partições), mas pode ser configurado como **CP** (consistência e tolerância a partições)

Yahoo! Cloud Serving Benchmarking (YCSB)

- Benchmark amplamente utilizado para sistemas de banco de dados NoSQL.
- O YCSB utiliza um modelo de dados simples baseado em chave-valor. O formato padrão do banco de dados possui 1 chave primária `YCSB_KEY` e um conjunto de dados `FIELD0`, `FIELD1`, ..., `FIELD9` que por padrão são tipo String.

Resultados do Cassandra

Tabela 1 - Tempos de benchmark no YCSB

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

Tabela 2 – Resultados do benchmark YCSB - Único nó

Operação	Operações	Latência Média (µs)	Latência Mín (µs)	Latência Máx (µs)	95% (µs)	99% (µs)
INSERT	10 000 000	1 222,82	197	148 607	1 961	5 299
READ	5 000 753	2 296,55	220	135 295	4 583	11 775
UPDATE	4 999 247	1 681,11	159	149 759	2 923	7 011

Tabela 3 – Resultados agregados do benchmark YCSB - 3 nós sem replicação

Operação	Operações	Latência Média (µs)	Latência Mín (µs)	Latência Máx (µs)	95% (µs)	99% (µs)
INSERT	10 000 000	703,01	159	139 135	1 084	2 669
READ	4 998 961	759,82	210	97 151	1 205	2 307
UPDATE	5 001 039	656,09	160	95 935	1 087	2 964

Tabela 4 – Resultados agregados do benchmark YCSB - 3 nós com replicação

Operação	Operações	Latência Média (µs)	Latência Mín (µs)	Latência Máx (µs)	95% (µs)	99% (µs)
INSERT	10 000 000	1 514,36	186	149 119	3 843	9 103
READ	5 000 119	2 334,83	268	125 119	5 563	11 191
UPDATE	4 999 881	1 170,27	182	114 431	3 099	5 983

Conclusão

Perguntas