UFSC-CTC-INE

INE 5600 – Bancos de Dados III

Bancos de Dados NewSQL

Bancos de Dados NewSQL

- Movimento relativamente recente pelo desenvolvimento de BDs SQL de alto desempenho visando o processamento OLTP eficiente de *Big Data*
  - Novos SGBDs também baseados na interface de acesso SQL
- Eles são também chamados de *newOLTP*
- BDs geralmente distribuídos

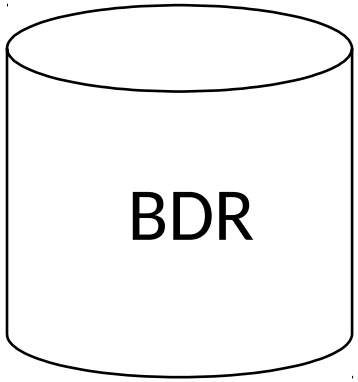
Diferenças em relação a BDDRs

- Maioria são BDs em memória
  - Utilizam novas tecnologias de HW para memória (ex.: NVRAM) com capacidade na casa de Tb e manutenção prolongada de dados em caso de falta de energia
- Adaptação de técnicas de gerenciamento de dados
  - *Scheduler*, *recovery*, particionamento
- Modelo lógico relacional e interface de acesso SQL, porém, o modelo físico não necessariamente é relacional
  - Exemplo: formato de armazenamento *chave-valor* é utilizado por alguns BDs NewSQL

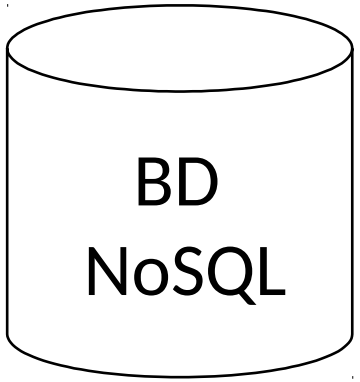
NewSQL - Motivação

- BDRs tradicionais não são uma boa escolha para *Big Data* pela sua dificuldade em escalar para grandes volumes de dados
- BDs NoSQL são uma boa escolha para aplicações *Big Data* que não se preocupam com consistência eventual (*read-intensive*)
- Mas e se a minha aplicação *Big Data* precisa de consistência forte?
  - BDs NoSQL não são uma boa escolha
    - Não suportam ACID e consultas complexas
    - Aplicação precisa escrever código para tratar isso!

NewSQL: “O Melhor de 2 Mundos”



- Interface *full* SQL
- Processamento OLTP
- Propriedades ACID



- Elasticidade
- Escalabilidade
- Processamento *Big Data*

NewSQL: BDR+ NoSQL

	BDR	BD NoSQL	BD NewSQL
Full SQL	suportado	não suportado	suportado
Arquitetura	tipicamente centralizada	tipicamente distribuída e na nuvem	tipicamente distribuída e na nuvem
Modelo de dados	relacional	chave-valor, colunar, documento, grafo	relacional *
Schemaless	não	sim	não
Propriedades	ACID	BASE	ACID
Escalabilidade Horizontal	não suportado	suportado	suportado
Complexidade das Consultas	alta	baixa	alta
Processamento de dados	dados simples e OLTP	Big Data e OLAP	Big Data, OLTP e OLAP

* existem algumas poucas soluções que se dizem NewSQL e adotam modelos NoSQL, fugindo desta tendência...

NewSQL - Aplicações

- Aplicações com alto processamento OLTP
  - Proliferação de aplicações Web em muitos tipos de dispositivos (*smartphones, tablets, notebooks, ...*) e usadas por um volume muito grande de pessoas
  - Exemplos: automação bancária, *e-commerce*, ...
- Aplicações de tempo real
  - Exigem dados consistentes que devem ser consumidos rapidamente
  - Exemplo: jogos *online multi-player*

NewSQL - Aplicações

- Aplicações voltadas a *Big Data analytics*
  - Consulta/análise de grande volume de dados e necessidade de resposta rápida
    - Exemplo: Bolsa de Valores (aplicações financeiras)
      - desejam computar tendências com base em movimentações contínuas de valores
  - Diferem dos tradicionais DWs, pois executam
    - Atualizações pontuais sobre dados indexados
    - Computação paralela massiva para análises em larga escala

NewSQL – Características Principais

1) Particionamento
2) Controle de Concorrência (*Scheduler*)
3) Replicação
4) *Recovery*

NewSQL – Características Principais

1) Particionamento
2) Controle de Concorrência (*Scheduler*)
3) Replicação
4) *Recovery*

NewSQL – Particionamento

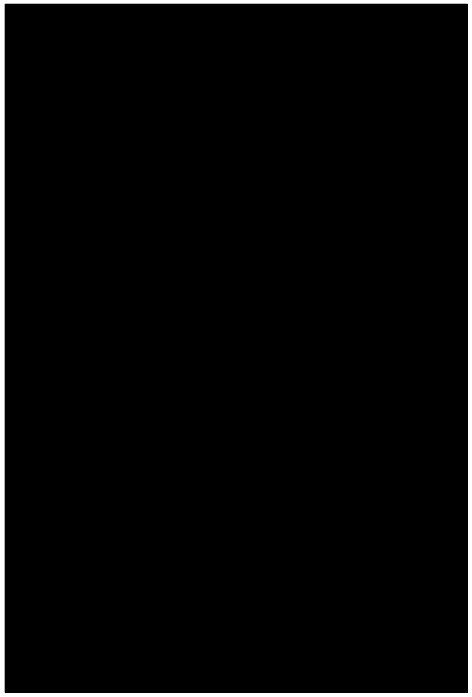
- Requisito importante para gerenciamento de *Big Data*
  - Gerenciamento tradicional centralizado não garante escalabilidade horizontal/elasticidade
- *Sharding*
  - Partição horizontal de várias tabelas em um nodo servidor de dados
  - Transações que desejam um certo dado irão encontrá-lo em um ou mais *shards* específicos, reduzindo o *overhead* de acesso a múltiplos nodos

NewSQL – Particionamento

- Particionamento é uma estratégia antiga aplicada em BDDRs, mas que conta atualmente com algumas vantagens
  - HW mais robusto e barato
  - Quando há mais demanda por *storage*, adiciona-se mais nodos ao *cluster* (escalabilidade horizontal), sem incorrer em reparticionamento complexo quando se tem um número limitado de nodos e se precisa reorganizar todos eles
    - Alguns SGBDs NewSQL suportam *live migration*
      - Capacidade de mover dados entre nodos sem interromper o transações e o controle das propriedades ACID
- Principais técnicas
  - *Range Partitioning*
  - *Hash Partitioning*

NewSQL – *Range Partitioning*

- Distribui os dados com base em intervalos da chave de partição das tabelas
- Um nodo servidor roteador gera a divisão e os demais *shards* são responsáveis por armazenar os dados e gerenciar as transações sobre eles
- Vantagens
  - Bom para *range queries*
  - Somente o nodo roteador se preocupa com a gerência do *sharding*
- Desvantagem
  - Gerenciamento centralizado do *sharding*



NewSQL – *Hash Partitioning*

- Universo de chaves é posicionado como um anel que é particionado em intervalos de tamanho aproximadamente igual entre os shards
- Uma função de *hash* distribui as chaves neste anel
  - Quanto melhor for a função de *hash*, mais uniformemente ela distribui as chaves nos intervalos
- Vantagem
  - Não necessita de um nodo roteador
- Desvantagem
  - Não é boa para *range queries*

NewSQL – Características Principais

1) Particionamento
2) Controle de Concorrência (*Scheduler*)
3) Replicação
4) *Recovery*

NewSQL – *Scheduler*

- BDDRs tradicionais usam técnicas de coordenação (1) centralizada ou (2) distribuída de transações distribuídas
  - BDs NewSQL optam geralmente pela técnica (2) que dá mais autonomia aos nodos para gerenciarem suas transações
- Técnicas baseadas em bloqueio (*lock*) são evitadas devido à complexidade de gerenciamento de *locks* e *deadlocks* distribuídos
  - Prefere-se técnicas que evitam *deadlocks*, como *Timestamp (TS)* e Multiversão (MVCC)

NewSQL – *TS Schedulers*

- *BD VoltDB*

  - Transações distribuídas são gerenciadas por um coordenador central e transações locais (TLs) por cada nodo
  - A execução das TLs é ordenada pelo TS e cada transação executa em completo isolamento (execução serial)
    - *VoltDB* é um BD em memória, por isso, executa transações com rapidez

NewSQL – *MVCC Schedulers*

- Técnica mais utilizada pelos BDs NewSQL
  - *Uma operação write(x) de uma transação T_k gera uma nova versão x' e, enquanto T_k está ativa, outras transações podem ler a versão antiga (x) do dado, evitando bloqueios de transações que desejam apenas ler o dado. Se T_k commitar, então x ← x'; senão fica x*
  - Vantagem: processa mais rápido que técnicas baseadas em *lock*
  - Desvantagens: eventual *garbage collection* para versões antigas e algoritmos de consenso (reconciliação)
- BD Clustrix
  - Técnica híbrida 2PL + MVCC
    - Define *locks* para atualização de dados, mas gera versões para permitir leitura do dado por outras transações

NewSQL – Características Principais

1) Particionamento
2) Controle de Concorrência (*Scheduler*)
3) Replicação
4) *Recovery*

NewSQL – Replicação

- Replicação garante maior disponibilidade de dados
  - Característica presente nos BDs NewSQL
  - Duas decisões de projeto
    - (1) Consistência Forte: a atualização de um dado X por uma transação T_k deve ser garantida em todas as réplicas que possuem X antes do *commit* de T_k;
    - (2) Consistência Fraca (ou Eventual): nem todas as réplicas que possuem X estão necessariamente atualizadas no *commit* de T_k;
  - (1) é adotado por NewSQL e (2) por NoSQL

NewSQL – Replicação

- A maioria dos BDs NewSQL adota a seguinte estratégia
  - X é atualizado inicialmente em um nodo (nodo *master*) e, posteriormente, o resultado da atualização é propagado para os demais nodos que possuem X (nodos *slaves*)
    - Vantagem: evita o reprocessamento da mesma operação sobre X em todas as réplicas

NewSQL – Características Principais

1) Particionamento
2) Controle de Concorrência (*Scheduler*)
3) Replicação
4) *Recovery*

NewSQL – *Recovery*

- BDs NewSQL tentam
  - Evitar a perda de atualizações em decorrência de falhas (o óbvio! ☺)
  - Minimizar o tempo de ociosidade do BD enquanto o SGBD se recupera de uma falha (*downtime*)
- *Recovery* em BDDRs tradicionais
  - Se um nodo A falha, um nodo B assume o controle das suas transações distribuídas e as *commita* (ou aborta), caso ele garanta a execução com sucesso (ou não) dessas transações nos demais nodos envolvidos
  - O nodo A, ao "voltar à vida", verifica a situação das suas transações distribuídas no *log* de B e se recupera

NewSQL – *Recovery*

- BDs NewSQL adotam essa técnica tradicional de *Recovery* distribuído, com eventuais adaptações
  - ZooKeeper
    - Serviço *open-source* da Apache de alto desempenho para coordenação distribuída e confiável
  - Adaptações de algoritmos de consenso tradicionais, como *Paxos* e *Raft*
    - Algoritmos paralelos de alto desempenho para apoiar as tarefas do novo gerente de transações B em decorrência de uma falha no nodo A

NewSQL – Principais Produtos

- Clustrix
- CockroachDB
- Google Spanner
- H-Store (pioneiro!)
- MemSQL
- NuoDB
- VoltDB