CEFET/RJ

Bancos de Dados Paralelos

Sistemas de banco de dados paralelos em grande escala são essenciais para armazenar esses volumes massivos, processar consultas complexas de apoio à decisão e oferecer alto throughput no processamento de transações.

Eduardo Ogasawara

eduardo.ogasawara@cefet-rj.br
https://eic.cefet-rj.br/~eogasawara

Paralelismo em Bancos de Dados

Particionamento de Dados

Os dados podem ser distribuídos entre múltiplos discos para permitir operações de entrada e saída paralelas, maximizando a eficiência do sistema.

Operações Paralelas

Operações relacionais como sort, join e agregação podem ser executadas simultaneamente, com cada processador trabalhando em sua própria partição de dados.

Consultas Simultâneas

Diferentes consultas podem ser executadas em paralelo, com o controle de concorrência gerenciando possíveis conflitos entre operações.

As consultas são expressas em linguagem de alto nível como SQL e traduzidas para álgebra relacional, tornando a implementação do paralelismo muito mais fácil e eficiente. Os bancos de dados são, portanto, candidatos naturais para arquiteturas paralelas.

Paralelismo de E/S

Conceito Fundamental

O paralelismo de E/S (entrada/saída) reduz significativamente o tempo necessário para recuperar relações do disco, através do particionamento estratégico dos dados em múltiplos discos físicos.

Particionamento Horizontal

No particionamento horizontal, as tuplas de uma relação são divididas entre diversos discos, de modo que cada tupla individual resida em exatamente um disco. Esta técnica permite acesso simultâneo aos dados, aumentando drasticamente a velocidade de recuperação.

Técnicas de Particionamento

Considerando n discos disponíveis

1

Rodízio (Round-Robin)

A i^a tupla inserida na relação é enviada ao disco calculado por i mod n. Esta técnica garante distribuição uniforme sem necessidade de análise dos dados.

2

Particionamento por Hash

Escolha um ou mais atributos como atributos de particionamento e defina uma função de hash h com intervalo de 0 a n-1. O resultado de h aplicado ao valor do atributo determina o disco de destino da tupla.

3

Particionamento por Intervalo

Define-se um vetor de particionamento que mapeia intervalos de valores de atributos para discos específicos, permitindo agrupamento natural dos dados.

Paralelismo por Intervalo

O particionamento por intervalo divide os dados, baseado em ranges específicos de valores de atributos. Por exemplo, em uma base de clientes, podemos particionar por faixas etárias: 18-30 anos no disco 1, 31-50 anos no disco 2, e acima de 51 anos no disco 3.

Esta técnica oferece vantagens significativas para consultas que envolvem intervalos de valores, pois apenas os discos relevantes precisam ser acessados. O agrupamento natural dos dados permite otimizações importantes no processamento de queries.

Comparação de Técnicas de Particionamento

Para avaliar a eficácia das técnicas de particionamento, devemos considerar três tipos fundamentais de acesso aos dados:

01

Varredura Completa

Leitura sequencial da relação inteira, processando todas as tuplas do início ao fim.

02

Consultas Pontuais

Localização de tuplas específicas usando busca associativa, como encontrar registros onde r.A = 25.

03

Consultas de Intervalo

Localização de tuplas cujos valores de atributo estão em um range especificado, por exemplo, $10 \leq r.A < 25$.

Vamos comparar como rodízio, hash e intervalo se comportam em cada cenário.

Análise de Particionamento por Rodízio

Vantagens

- Ideal para varredura sequencial completa da relação
- Distribuição uniforme: todos os discos têm aproximadamente o mesmo número de tuplas
- Balanceamento perfeito de carga entre os discos durante a recuperação
- Simplicidade de implementação

Limitações

As consultas de intervalo são extremamente difíceis de processar com eficiência. Como não há agrupamento natural dos dados, as tuplas relevantes ficam espalhadas por todos os discos do sistema, exigindo acesso a múltiplos discos mesmo para intervalos pequenos.

Análise de Particionamento por Hash

Acesso Sequencial

Excelente para varredura completa. Assumindo uma boa função de hash e atributos de particionamento formando uma chave, as tuplas são distribuídas uniformemente entre os discos, garantindo balanceamento efetivo de carga.

Consultas Pontuais

Desempenho ótimo para buscas sobre o atributo de particionamento. É possível pesquisar um único disco, deixando os outros disponíveis para responder a outras consultas. Índices locais tornam buscas e atualizações mais eficientes.

Consultas de Intervalo

Desempenho ruim devido à ausência de agrupamento. Os valores em um intervalo são distribuídos aleatoriamente entre os discos, exigindo acesso a múltiplos discos mesmo para ranges pequenos.

Análise de Particionamento por Intervalo

O particionamento por intervalo oferece agrupamento natural dos dados baseado nos valores dos atributos de particionamento, proporcionando benefícios únicos para diversos tipos de consultas.

Vantagens Principais

- Excelente para acesso sequencial com boa distribuição
- Ótimo para consultas pontuais: apenas um disco precisa ser acessado
- Superior para consultas de intervalo: poucos discos são necessários
- Discos não utilizados ficam disponíveis para outras queries

Limitação Importante

Quando muitos blocos precisam ser recuperados de poucos discos, o paralelismo potencial no acesso é desperdiçado. Este é um exemplo clássico de distorção de execução, onde o benefício do paralelismo é limitado pela concentração de dados.

Distorção em Particionamento

A distribuição de tuplas entre discos pode ser distorcida, resultando em alguns discos sobrecarregados enquanto outros permanecem subutilizados. Este desequilíbrio compromete seriamente o desempenho paralelo.

Distorção de Valor de Atributo

Ocorre quando determinados valores aparecem com alta frequência nos atributos de particionamento. Todas as tuplas com o mesmo valor acabam na mesma partição, criando um hotspot.

Pode ocorrer tanto com particionamento de intervalo quanto com particionamento de hash, especialmente quando há valores dominantes nos dados.

Distorção de Partição

No particionamento por intervalo, um vetor de particionamento mal escolhido pode atribuir muitas tuplas a algumas partições e poucas a outras.

Menos provável no particionamento de hash, desde que uma boa função de hash seja escolhida, pois a distribuição tende a ser mais uniforme.

Tratamento da Distorção usando Histogramas

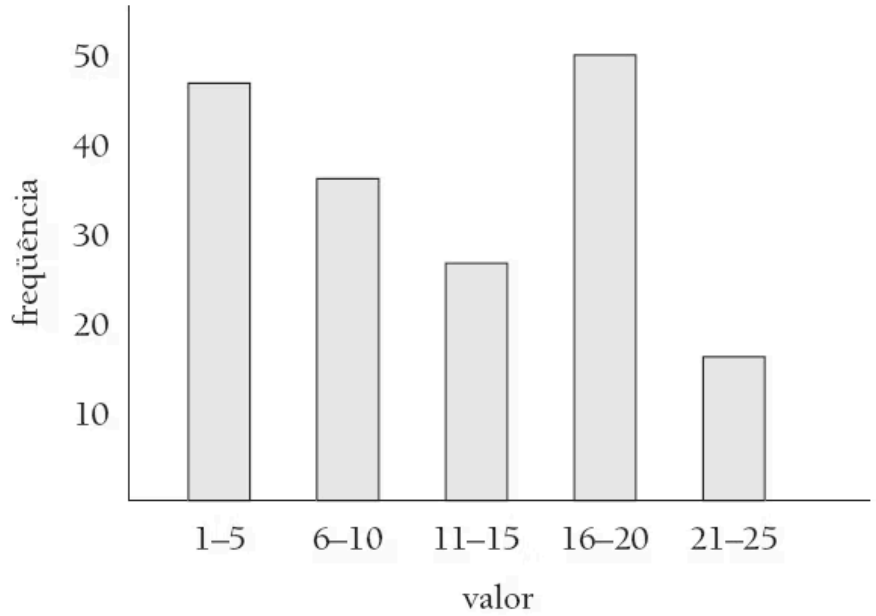
Construção do Vetor de Particionamento

Um vetor de particionamento balanceado pode ser construído a partir de um histograma da distribuição dos dados, seguindo um padrão relativamente simples.

A técnica assume distribuição uniforme dentro de cada intervalo do histograma, permitindo divisões mais precisas das partições.

Métodos de Construção

- Varredura completa da relação para análise detalhada
- Amostragem de tuplas para estimativa rápida
- Amostragem de blocos para reduzir overhead



Particionamento de Processador Virtual

A distorção no particionamento de intervalo pode ser tratada de forma elegante usando a técnica de particionamento de processador virtual, que oferece uma solução sofisticada para o problema de balanceamento de carga.

Criar Partições Virtuais

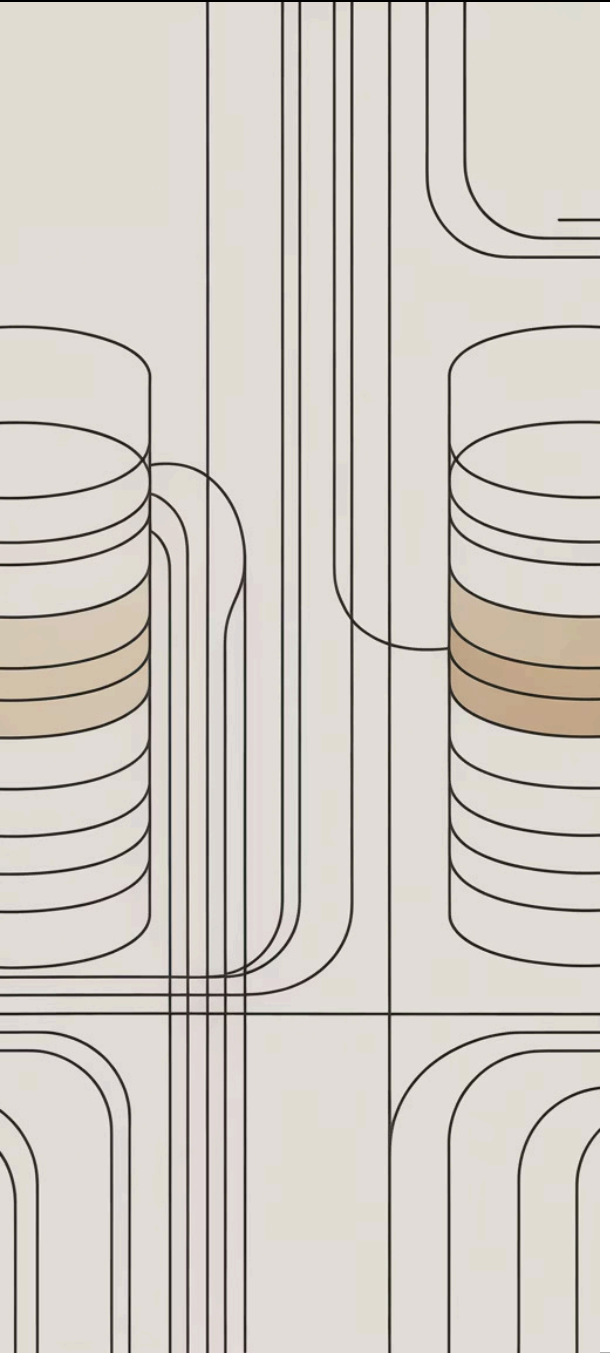
Crie uma grande quantidade de partições virtuais, tipicamente de 10 a 20 vezes o número de processadores físicos disponíveis no sistema.

Atribuir Processadores

Atribua processadores virtuais às partições usando padrão de rodízio ou baseado no custo estimado de processamento de cada partição virtual.

Distribuir Carga

Se qualquer partição for distorcida, a distorção se espalha por diversas partições virtuais, que são distribuídas entre processadores, garantindo trabalho uniforme.



Paralelismo Entre Consultas

O paralelismo entre consultas permite que múltiplas queries e transações sejam executadas simultaneamente, aumentando significativamente o throughput do sistema. Esta é a forma mais direta de paralelismo para implementar.

Arquiteturas Compartilhadas

Em bancos de dados paralelos de memória compartilhada, a implementação é relativamente simples, pois até mesmo sistemas sequenciais já suportam processamento concorrente básico.

Desafios em Arquiteturas Distribuídas

Em arquiteturas de disco compartilhado ou nada compartilhado, a complexidade aumenta: bloqueio e logging precisam ser coordenados via passagem de mensagens, dados em buffers locais podem estar desatualizados, e a coerência de cache deve ser mantida.

Paralelismo Intraconsulta

A execução de uma única consulta em paralelo através de múltiplos processadores e discos é crucial para acelerar consultas de longa duração e melhorar significativamente o tempo de resposta.

Paralelismo Intraoperação

Paraleliza a execução de cada operação individual dentro da consulta. Esta abordagem divide o processamento de uma única operação entre múltiplos processadores.

Paralelismo Entre Operações

Executa diferentes operações de uma expressão de consulta simultaneamente, aproveitando a independência entre operações sequenciais.

O paralelismo intraoperação geralmente se expande melhor com o aumento do número de processadores, pois o número de tuplas processadas por cada operação normalmente é muito maior que o número de operações em uma consulta.

Classificação Paralela

A operação de classificação (sort) é fundamental em bancos de dados e pode ser significativamente acelerada através de técnicas paralelas. O processo divide o conjunto de dados entre múltiplos processadores, onde cada um classifica sua porção localmente.

A estratégia mais comum é o particionamento por intervalo dos dados, seguido de classificação local em cada processador. Os ranges são escolhidos para garantir que todos os elementos em uma partição sejam menores que os elementos da partição seguinte, eliminando a necessidade de uma etapa final de merge global.

O desafio principal é escolher pontos de divisão que resultem em partições balanceadas. Técnicas de amostragem e histogramas são frequentemente usadas para determinar ranges apropriados que distribuam uniformemente a carga de trabalho.

Junção Paralela

A operação de junção (join) é uma das mais custosas em bancos de dados e se beneficia enormemente do paralelismo. O objetivo é testar pares de tuplas para verificar se satisfazem a condição de junção e, em caso positivo, acrescentá-los à saída.

Divisão de Pares

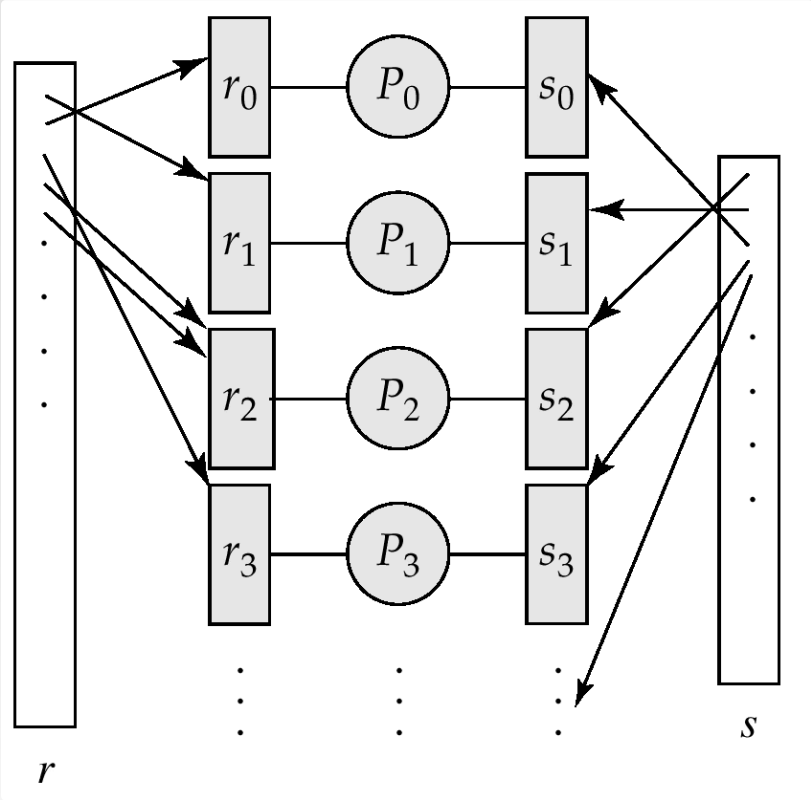
Algoritmos paralelos dividem os pares a serem testados entre vários processadores

Processamento Local

Cada processador calcula sua parte da junção de forma independente

Consolidação

Resultados de todos os processadores são coletados para formar o resultado final



Junção Particionada

A junção particionada é uma técnica sofisticada que utiliza particionamento de hash para distribuir as tuplas das relações de entrada entre os processadores. As duas relações R e S são particionadas usando a mesma função de hash nos atributos de junção.

Esta abordagem garante que tuplas correspondentes de R e S sejam enviadas ao mesmo processador. Cada processador então executa a junção localmente sobre suas partições, sem necessidade de comunicação com outros processadores durante o processamento.

A eficiência dessa técnica depende criticamente da qualidade da função de hash, que deve distribuir uniformemente as tuplas para evitar desbalanceamento de carga entre processadores.

Paralelismo Entre Operadores

O paralelismo entre operadores, também conhecido como paralelismo em pipeline, permite que diferentes operações de uma consulta sejam executadas simultaneamente em um fluxo contínuo de dados.

Nesta abordagem, enquanto um operador processa um conjunto de tuplas, ele passa os resultados intermediários diretamente para o próximo operador, que já pode começar seu processamento. Isso elimina a necessidade de materializar resultados intermediários em disco, economizando tempo de E/S e espaço de armazenamento.

Por exemplo, em uma query com seleção seguida de projeção e então junção, as tuplas podem fluir através desses operadores continuamente, com cada estágio processando diferentes tuplas simultaneamente. Esta técnica é especialmente efetiva quando os operadores têm tempos de processamento similares.

Limitações do Paralelismo Canalizado

Embora útil, o paralelismo em pipeline apresenta limitações importantes que restringem sua aplicabilidade e escalabilidade em sistemas de banco de dados paralelos de grande escala.

Escalabilidade Limitada

Útil com pequena quantidade de processadores, mas não se expande bem além disso. As cadeias de pipeline raramente conseguem tamanho suficiente para utilizar efetivamente dezenas ou centenas de processadores.

Operadores Bloqueantes

Impossível canalizar operadores que não produzem saída até que todas as entradas tenham sido processadas, como agregação completa (SUM, AVG) e operações de ordenação (ORDER BY).

Distorção de Execução

Pouco ganho é obtido em casos frequentes onde o custo de execução de um operador é muito maior que os outros, criando um gargalo que limita o throughput de todo o pipeline.

Por essas razões, o paralelismo intraoperação geralmente oferece melhores oportunidades de escalabilidade e desempenho em sistemas modernos.

Referências

Elmasri & Navathe

Fundamentals of Database Systems

Pearson, 2016

Referência abrangente sobre fundamentos de sistemas de bancos de dados, cobrindo aspectos teóricos e práticos.

Korth, Sudarshan & Silberschatz

Database System Concepts

McGraw-Hill, 2019

Texto fundamental que serviu como base para a maioria dos exemplos apresentados nesta apresentação.

Özsu & Valduriez

Principles of Distributed Database Systems

Springer Nature, 2019

Obra especializada em sistemas de bancos de dados distribuídos, essencial para compreensão avançada.

Nota: Os conceitos e exemplos apresentados baseiam-se principalmente na literatura clássica de sistemas de bancos de dados, em especial *Database System Concepts* e *Fundamentals of Database Systems*.