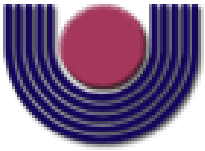


UNIOESTE – Universidade Estadual do Oeste do Paraná

CENTRO DE CIÊNCIAS EXATAS E TECNOLÓGICAS

Colegiado de Informática

Curso de Bacharelado em Informática

CASCAVEL

2009

Allan Roger Bello, Willian Tamagi

Banco de Dados Paralelos

.....

Trabalho de graduação da disciplina de Banco de Dados I, 4º ano do curso de Bacharelado em Informática, do Centro de Ciências Exatas e Tecnológicas da Universidade Estadual do Oeste do Paraná - Campus de Cascavel

Orientador: Carlos José Maria Olguín

CASCAVEL

2009

Allan Roger Bello, Willian Tamagi

Bancos de Dados Paralelos

.....

Monografia apresentada como requisito parcial para obtenção de nota na disciplina de bando de dados , pela Universidade Estadual do Oeste do Paraná, Campus de Cascavel, aprovada pela Comissão formada pelo professor:

_____

Prof. Carlos José Maria Olguín

Colegiado de Informática, UNIOESTE

Cascavel, 22 de Setembro de 2009.

Sumário

Resumo

Arquiteturas de máquinas paralelas base de dados têm evoluído a partir do uso de hardware exótico para uma arquitetura de software de fluxo de dados paralelo baseado em hardware convencional compartilhada nada. Estes novos modelos proporcionar aceleração impressionante e scaleup ao processar consultas de banco de dados relacional. Este artigo revisa as técnicas utilizadas por esses sistemas, e as pesquisas comercial vigente e investigação de sistemas.

Palavras-chave: Banco de dados, paralelismo, paralelo, SBDP.

Capítulo 1

Introdução

Enquanto há várias direções nas quais sistemas de bancos de dados modernos estão evoluindo, novas aplicações exigem grandes volumes de processamento e pesquisas complexas no banco de dados, devendo oferecer tempo de resposta satisfatório. Algumas aplicações utilizam dados armazenados em dois ou mais bancos de dados (fontes de informação) e constroem a partir deles um banco de dados maior, contendo informação de todas as fontes, de forma que os dados possam ser consultados como uma unidade.

Tendo em vista esta forma de manipulação, os bancos de dados centralizados numa única máquina ou disco já não são a melhor solução para uma diversidade de aplicações. Para várias famílias de aplicações, é importante que a informação esteja facilmente acessível, favorecendo a sua disponibilidade. Para permitir que esta situação se tornasse realidade, foram criados os bancos de dados distribuídos.

No decorrer desta evolução, sentiu-se a necessidade de bancos de dados que, mesmo com o armazenamento de grandes volumes de informação permitissem um processamento eficiente sobre eles, sem a necessidade de aquisição de poderosas arquiteturas de hardware. Desta forma, disseminou-se no mercado a tecnologia dos bancos de dados paralelos.

Neste capítulo, será feita a discussão destas duas tecnologias, os bancos de dados paralelos e distribuídos, enfatizando os conceitos necessários para o tratamento e entendimento do presente trabalho.

Capítulo 2

Bancos de Dados Paralelos

Um sistema de banco de dados paralelo implica na distribuição dos dados do banco de dados pelos vários nós do sistema de uma maneira que permita transparência plena.

Arquiteturas de bancos de dados paralelos têm sido apresentadas. Formas de distribuição de dados e sistemas de processamento e otimização de consultas têm sido desenvolvidos.

A adoção do sistema relacional de dados é uma das explicações para o sucesso de sistemas de banco de dados paralelo, buscas relacionais são criadas pensando no paralelismo. Cada operador produz uma nova relação, então os operadores podem ser decompostos em requisições paralelas. Através do envio da resposta de saída de um operador na entrada de requisições paralelas. Através do envio da resposta de saída de um operador na entrada de requisição de outro operador, os dois operadores podem trabalhar em séries através do paralelismo de pipeline. Particionando a requisição em vários processadores e memórias, um operador pode também ser dividido em vários operadores independentes, cada um tratando parte de requisição. Esse particionamento de dados e execução é a base do paralelismo particionado

O sistema de troca de dados para criação de sistemas de banco de dados precisa de um sistema operacional baseado em troca de mensagens entre cliente e servidor, para inter-relacionar os processos paralelos executando as operações relacionais. Essa arquitetura também depende de um rede de alta-velocidade para conectar os processadores, configuração essa que atualmente se tornou a base para os PCs. Esse sistema cliente-servidor é uma excelente base para a tecnologia de sistemas de banco de dados distribuídos.

Os criadores de mainframes encontraram dificuldade em criar máquinas com capacidade suficiente para garantir a demanda dos bancos de dados relacionais servindo um grande número de usuários ou buscando bancos de dados com terabytes de informação. Enquanto isso, sistemas multiprocessados baseados em microprocessadores rápidos e baratos ficaram disponíveis através de empresas como a Encore, Intel, NCR, nCUBE, Sequent, Tandem, Teradata e Thinking Machines. Essas máquinas provinham mais poder de processamento total do que seus mainframes concorrentes a um preço mais baixo. Sua arquitetura modular permitiu aos sistemas crescerem acrescentando-se memória e discos para facilitar o processamento de um trabalho qualquer em paralelo.

Surge então outra arquitetura baseada em share-nothing (compartilhar nada), na qual cada processador se comunica com os outros apenas enviando mensagens através de uma rede interconectada. Nesse tipo de sistemas, as tuplas de cada relacionamento no banco de dados são particionados através dos discos diretamente ligados a cada um dos processadores. O particionamento permite que vários processadores varram grandes relações em paralelo sem

necessitar de nenhum sistema de E/S exótico. Está arquitetura surgiu pioneiramente na Teradata no final dos anos 70 e também em inúmeros projetos de pesquisa.

2.1 Arquiteturas de Bancos de Dados Paralelos

O sistema de banco de dados ideal deveria ter um único e infinitamente rápido processador com uma infinita memória – e seria infinitamente barato (de graça). Dado essa máquina não haveria necessidade de aumento de velocidade, escalabilidade, ou paralelismo. Infelizmente, a tecnologia não está criando tal máquina – mas está chegando perto.
Então o desafio é criar um processador infinitamente rápido através de infinitos processadores de velocidade finita, e criar uma memória infinitamente grande com infinita área de troca de infinitas memórias de velocidade e armazenamento finito. Isso soa trivial matematicamente; mas na prática quando um novo processador é adicionado à maioria das implementações de computadores, ele diminui um pouco da velocidade de todos os outros processadores. Se essa interferência é de 1%, um sistema de mil processadores teria 4% do poder efetivo de um sistema com um único processador de mesma velocidade.
Stonebraker sugere as seguintes taxonomias de design para sistemas paralelos:

- Memória compartilhada: Todos os processadores compartilham a mesma memória e os mesmos discos.
- Discos compartilhados: Cada processador tem uma memória própria, mas tem acesso a todos os discos.
- Sem compartilhamento: Cada memória e disco são próprios de um processador que atua como servidor dos dados que possui.
- Hierárquico: Cada nó pode ser considerado como um sistema independente.

2.1.1 Memória Compartilhada

Os processadores e os discos acessam uma memória em comum, normalmente, por meio de cabo ou por meio de rede de interconexão

- Vantagem: extrema eficiência na comunicação entre processadores
- Desvantagem: a arquitetura não é adequada ao uso de mais de 32 ou 64 processadores
  - Exemplos: multiprocessadores simétricos (Sequent, Encore) e alguns mainframes (IBM3090, Bull's DPS8)

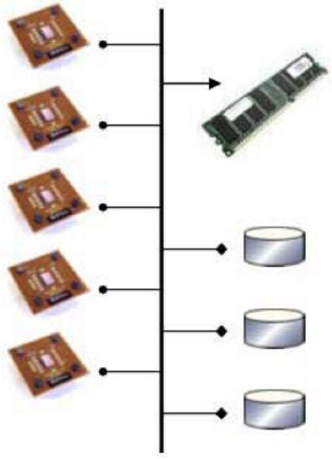


Figura 1: Arquitetura de um SBDP com memória compartilhada

2.1.2 Discos Compartilhados

Todos os processadores podem acessar diretamente os discos através de uma rede de conexão, mas cada processador possui uma memória privada.

- Vantagens: o acesso à memória não representa um gargalo; é um modo barato de aumentar a tolerância a falhas
- Desvantagem: é novamente o grau de crescimento

o Exemplos: IBM Sysplex e Digital VAXclusters rodando Rdb (Oracle Rdb)

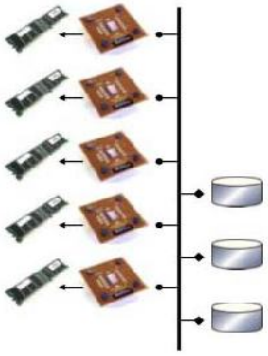


Figura 2: Arquitetura de um SBDP com discos compartilhados

2.1.3 Sem Compartilhamento

Cada equipamento de um nó consiste em um processador, uma memória e discos.

- Vantagem: suporte a um grande número de processadores
- Desvantagem: comunicação entre processadores é o fator limitante, devido a necessidade de acesso a dados não locais

o Exemplos: nCUBETeradata's DBC, Tandem, Intel's Paragon, NCR's 3600 e 3700

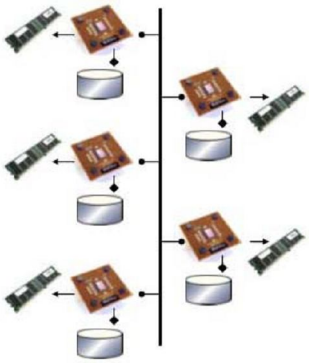


Figura 3: Arquitetura de um SBDP sem compartilhamento

2.3. Hierárquico

Este modelo combina características de várias arquiteturas anteriores, reduzindo a necessidade e complexidade da comunicação entre processadores.

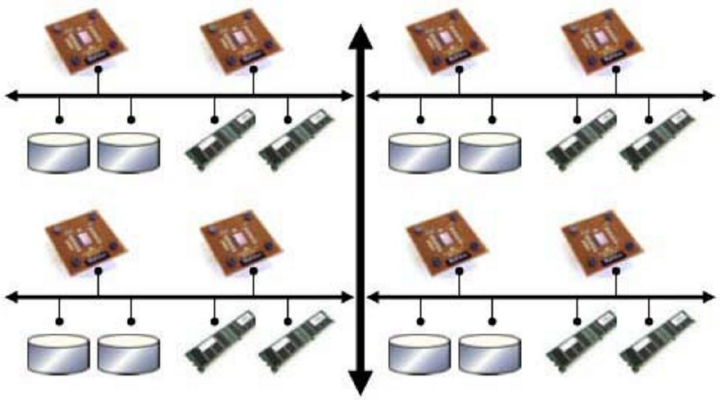


Figura 4: Arquitetura de um SBDP hierárquico

Capítulo 3

Implementando o Paralelismo em Bancos de Dados.

O paralelismo pode ser utilizado de 3 (três) formas:

- Na entrada e saída de dados (E/S)
- No processamento de consultas
- No processamento de operações individuais

Aumentando a escala e o desempenho do sistema, oferecendo um maior e mais rápido processamento das transações.

Entretanto, torna o sistema mais exigente quanto ao hardware, podendo ocasionar uma maior quantidade de falhas. Uma solução para tal contratempo seria a replicação dos dados, para um controle mais eficaz da consistência.

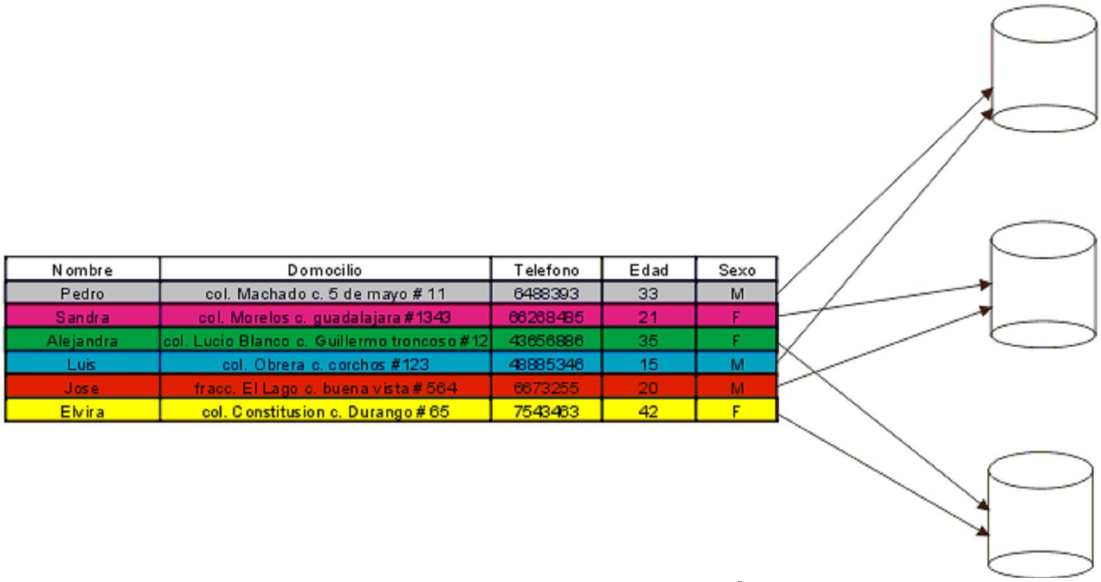
3.1 Paralelismo na entrada e saída de dados (E/S)

O paralelismo de E/S tenta reduzir o tempo necessário para recuperar relações do disco por meio do particionamento dessas relações em múltiplos discos.
Nesta forma as operações podem ser executadas em paralelo se cada uma acessar um dispositivo, assim sendo, cada processador pode trabalhar com os dados de uma partição. Um plano de execução em paralelo pode ser criado ao otimizar a operação
Existem três tipos de particionamento de dados para obtenção de paralelismo de E/S:

- Horizontal
- Vertical
- Misto

3.2 Particionamento Horizontal

A forma mais comum de particionamento de dados em um ambiente de banco de dados é o particionamento horizontal:



Nombre	Domocilio	Telefono	Edad	Sexo
Pedro	col. Machado c. 5 de mayo # 11	6488393	33	M
Sandra	col. Morelos c. guadalajara #1343	66268485	21	F
Alejandra	col. Lucio Blanco c. Guillermo troncoso #12	43656886	35	F
Luis	col. Obrera c. corchos #123	48885346	15	M
Jose	fracc. El Lago c. buena vista #564	6673255	20	M
Elvira	col. Constitusion c. Durango #65	7543463	42	F

Figura 5: Particionamento Horizontal

As duplas de uma relação são divididas entre muitos discos, tal que cada dupla resida em um disco diferente

3.3 Técnicas de Particionamento

Existem três estratégias básicas de particionamento, para uma melhor implementação do “trabalho” a ser executado. São elas:
Considerando **n** discos, D_0, D_1, D_2, ..., D_{n-1}, entre os quais os dados devem ser particionados, explicaremos as estratégias.

Round-Robin (circular)

- A relação é percorrida em qualquer ordem e a i-ésima tupla é enviada ao disco numerado como $D_{i \bmod n}$.
- Cada nova dupla é colocada em um dispositivo diferente, distribuindo uniformemente entre os discos.

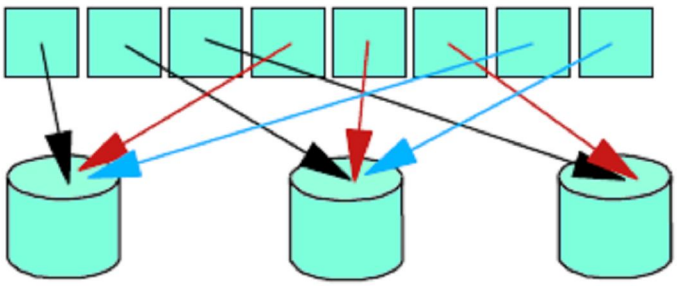


Figura 6: Round-Robin

Particionamento Hash

- Um ou mais atributos do esquema de relação dado são designados como atributos de particionamento.
- Uma função Hash é escolhida em uma faixa entre { 0, 1, ..., n-1 }.
- Cada tupla da relação original é separada pelo atributo de particionamento. Se a função Hash retorna **i**, então a tupla é alocada no disco D_i.

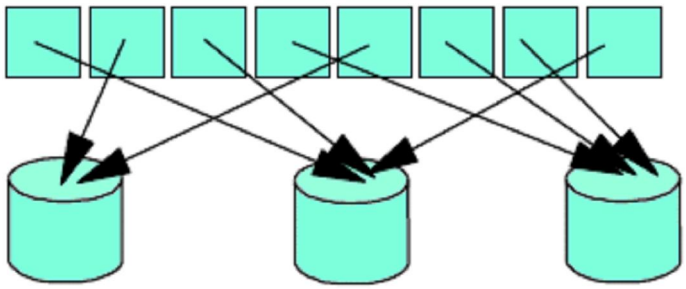


Figura 7: Posicionamento Hash

Particionamento por faixa

- Distribui faixas contíguas do valor de um atributo para cada disco.
- Um atributo de particionamento **A** é escolhido como um vetor de particionamento.
- Seja a seqüência $[v_0,v_1,...,v_{n-2}]$ denotando o vetor de particionamento, tal que, se $i<j$, então $v_i<v_j$.
    - Considere uma tupla **t**, tal que t[A] = x.
    - Se $x<v_0$, então **t** é colocada no disco D_0.
    - Se $x = v_{n-2}$ então **t** é colocada no disco D_{n-1}.
    - Se vi = x < vi+1, então **t** é colocada no disco Di+1.

3.4 Comparação de Técnicas de Particionamento

Uma vez particionada a relação, podemos recuperá-la usando vários tipos de acesso aos dados:

- Percorrer a relação inteira

- Localizar uma dupla associativamente (por exemplo, nome_empregado = “João”)
- Localizar todas as duplas, tal que o valor de um dado atributo permaneça entre ma faixa especificada

(por exemplo, 10000 < salário < 20000)

Round-Robin

• É ideal para aplicações que precisam ler a relação inteira, seqüencialmente, em cada consulta. Entretanto, tanto consultas pontuais, como por faixas têm processamento complexo, já que cada um dos *n* discos precisará participar da busca.

Hash

• É mais adequado consultas pontuais baseadas no atributo de particionamento.

• É útil para varreduras seqüenciais em uma relação inteira. O número de tuplas em cada um dos discos é o mesmo, sem muita variação, portanto, o tempo usado para percorrer a relação é 1/**n** do tempo necessário para percorrer a relação em único disco.

• Não é muito adequado para consultas pontuais sobre demais atributos de não-particionamento

• Também não é muito adequado para responder consultas sobre e faixas de dados, já que normalmente as funções Hash não preservam proximidade entre as faixas.

Por Faixa

• É bastante adequado para consultas pontuais e por faixas sobre atributos de particionamento.

- Um ou poucos discos precisam ser utilizados
- Os outros discos ficam livres para outros acessos
- Eficiente se as tuplas do resultado estiverem em poucos blocos de disco
- Se muitos blocos precisarem ser lidos, há desperdício de paralelismo pois poucos discos serão utilizados

• Em buscas pontuais recorremos ao vetor de particionamento para localizar o disco no qual a dupla reside

• Para consultas por faixas, recorremos ao vetor de particionamento a fim de encontrar a faixa de disco na qual as tuplas podem residir. Em ambos os casos, reduz-se a busca a exatamente aqueles discos que podem ter quaisquer duplas de interesse

Desbalanceamento

Quando uma alta porcentagem de duplas são colocadas em algumas partições e poucas duplas são colocadas nas restantes, ocorre o que chamamos de desbalanceamento.

Formas de desbalanceamento:

• Desbalanceamento de valor de atributo: muitas duplas possuem valores idênticos para atributo de particionamento

• Desbalanceamento de partição: Os critérios de particionamento concentram muitas duplas em poucas partições, menos provável no hash do que no particionamento por faixas, se a função do hash dor boa o suficiente

3.5 Particionamento Vertical

Os campos de uma relação se particionam entre os discos, onde cada campo reside em um ou mais discos. A partição vertical de uma relação **R** produz **R**1, **R**2, ..., **R**R, cada um dos quais é um subconjunto dos atributos de **R.** O objetivo consiste em dividir a relação em um conjunto de relações menores que o original, de tal forma que se minimize o tempo de execução das aplicações que implementam esses fragmentos.

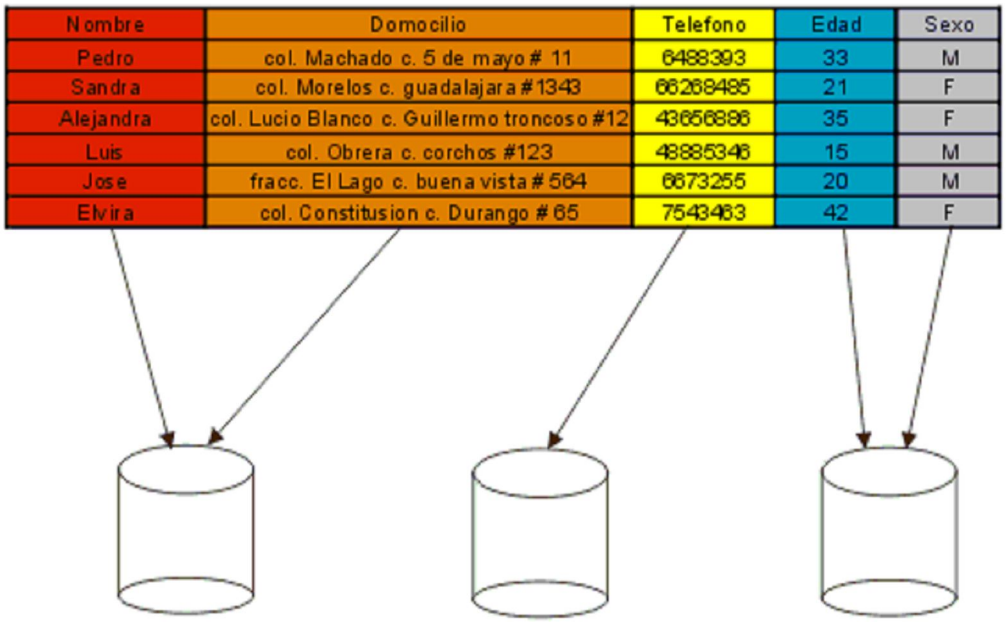


Figura 8: Particionamento Vertical

A partição vertical resulta ser mais difícil que a horizontal devido a necessidade de análises estatísticas sobre os acessos realizados a cada atributo da relação. Deve-se auxiliar replicando as chaves primárias da relação para poder reconstruir a relação original.

3.6 Particionamento Misto

Em alguns casos o uso do particionamento horizontal ou vertical não se apresenta satisfatoriamente para a aplicações que acessão a base de dados, de tal forma que se faz necessário o uso da partição mista.

Existem dois tipos de particionamento misto:

• HV: Faz-se uma partição vertical, e posteriormente, sobre os fragmentos resultantes se aplica uma partição horizontal.

• VH: Faz-se uma partição vertical sobre os conjuntos de tuplas resultantes da aplicação de uma partição horizontal.

Nesse tipo de partição necessitaremos dados quantitativos sobre a base de dados, as aplicações que funcionam sobre ela, a rede de comunicações, as características de processo e o limite de armazenamento de cada local de rede.

3.7 Paralelismo em Consultas

O processamento de consultas em arquiteturas paralelas deve considerar o tipo de compartilhamento de discos e memória existente e a fragmentação e alocação de dados empregada.

Há dois tipos de paralelismo possível para o processamento de consultas em um bando de dados paralelos:

- Inter-consultas
- Intra-consultas

3.7.1. Inter-consultas

Esta e a alternativa mais simples. Cada consulta submetida pelo usuário é executada totalmente em um único processador. O tempo de processamento de uma certa consulta é idêntico ao tempo em um servidor mono processado, pois a consulta é monoprocessada. Por conseqüência, os processadores devem ser de grande capacidade, para não inviabilizar o sistema. A vantagem é que o gerenciamento de tarefas é bastante simples.

Consultas ou transações diferentes são executadas em paralelo umas com as outras. A principal aplicação do paralelismo inter-consultas é melhorar o sistema de processamento de transações processadas por segundo do paralelismo inter-consultas é melhorar o sistema de processamento de transações processadas por segundo. Os processadores têm de realizar algumas tarefas como bloqueio e log (registro diário), de forma coordenada e isso exigem que

troquem mensagens entre si, além de assegurar que dois processadores não atualizem o mesmo dado, de modo independente, ao mesmo tempo.

É necessária a *coerência de cache*, que consiste em garantir que um processador, ao acessar ou atualizar dados, tenha a última versão dos dados em sua área de buffer. Algumas soluções para tal problema são:

- Antes de qualquer acesso para leitura ou escrita de uma página, uma transação bloqueia a página e lê a sua cópia mais recente no disco compartilhado.
- Antes de uma transação liberar um bloqueio exclusivo em uma página, ela descarrega a página no disco compartilhado.

3.7.2 Paralelismo intra-consultas

Partes de uma consulta são executadas em paralelo nos diversos processadores e discos, o que diminui o tempo de resposta das consultas.

Pode-se fazer planos de execução em forma de árvores, e cada ramo pode ser processado em paralelo. Entre as operações pode-se utilizar *Pipelining*, assim, a saída de uma operação é a entrada da outra. Caso não seja possível fazer o *pipelining*, utiliza-se a memória e os discos compartilhados para a troca de dados entre os processadores.

3.8 Paralelismo no processamento de operações individuais

A avaliação paralela de operações apresenta alguns custos, tais como:

- Particionamento de E/S entre diversos discos
- Particionamento de CPU entre diversos processadores
- Custos de inicialização em diversos processadores
- Desbalanceamento da distribuição do trabalho entre os processadores
- Retenção de recursos, resultando em atrasos

- Custo de montagem do resultado final, devido à transmissão de resultados parciais a partir de cada processador

Existem duas formas de paralelismo para o processamento de operações em um banco de dados, as quais podem ser utilizadas simultaneamente em um SGBDP.

3.8.1 Paralelismo inter-operação

As operações de uma consulta são executadas em paralelo. Como uma consulta é dividida em várias operações mais simples, a alocação de processadores é definida para cada operação, o que reduz o tempo para a consulta. Ao alocar as operações nos processadores, uma operação produz um resultado parcial que é a entrada para a outra, preferencialmente, no mesmo processador, a fim de reduzir a comunicação entre processadores.

Este método pode ser implementado de duas formas:

- **Paralelismo Independente:** As operações de uma consulta não dependem necessariamente umas das outras, assim, as operações independentes são executadas em paralelo por processadores diferentes.

- **Paralelismo Pipeline:** Algumas operações não são independentes, assim cada operação é executada por um processador. As tuplas produzidas por uma operação são passadas para as operações que precisam delas.

3.8.2 Paralelismo intra-operação

Uma operação é dividida em várias partes, sendo cada uma delas executada por um processador. Os algoritmos de cada operação podem ser paralelizados, possibilitando que uma operação complexa seja realizada por vários processadores, o que reduz o tempo de resposta da operação. Entretanto, há a necessidade de desenvolver algoritmos e otimizá-los para cada tipo de arquitetura e particionamento de dados utilizado.

3.8.2.1 Implementações do método de paralelismo intra-operação

Algumas implementações utilizadas no método de paralelismo intra-operação são:

- **Seção paralela:** Cada processador procura na sua partição dos dados as tuplas nas quais a condição de seleção é válida. A tabela pode ser particionada por faixa ou *hash* para simplificar o método, já que não é preciso usar todos os processadores.

- **Classificação em paralelo:**
  – *Algoritmo de classificação por faixas:*
    - Particionar a tabela por faixas com base no atributo de classificação
    - Ordenar cada partição independentemente
  – *Algoritmo de sort-merge externo paralelo:*
    - Supondo que a tabela já foi particionada em vários discos usando qualquer método
    - *Sort:* Cada processador ordena uma partição
    - *Merge:* Partições já ordenadas são particionadas novamente por faixas e enviadas aos processadores, que as unem às outras partições recebidas e as classificam
- **Eliminação de duplicatas em paralelo:** Pode ser feita na classificação das tuplas ou nas faixas ou *hashs* por cada processador.

- **Projeção paralela:** Efetuada pelos processadores à medida que as tuplas são lidas em paralelo dos discos.

- **Agregação:** Particionar a relação em faixa ou *hash* utilizando os atributos de agrupamento e computar os valores agregados em cada processador.

3.8.2.2 Junção paralela

Algoritmos dividem entre os processadores os pares de tuplas a serem testados na junção, de forma que os pares de tuplas obtidos dos processadores para os quais a condição de junção é válida são reunidos para determinar o resultado final da junção.

- **Algoritmo de junção particionada:** É utilizado em junções naturais e junções de igualdade.
  - A mesma função de faixa ou *hash* deve ser usada para particionar as tabelas.
  - Cada processador faz a junção das partições usando um algoritmo de junção seqüencial.

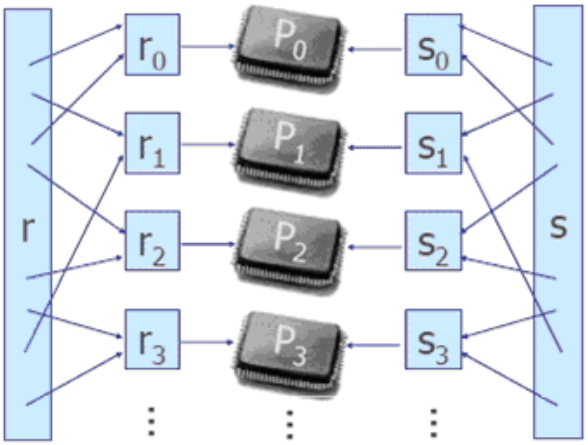


Figura 10: Algoritmo de junção particionada

- **Algoritmo de junção por fragmentação e replicação:** É utilizado em junções que não são naturais e nem de igualdade.
  - As duas tabelas são fragmentadas em *m* e *n* partições, respectivamente.
  - Cada processador executa a junção entre uma partição da primeira tabela e outra da segunda, usando qualquer algoritmo de junção local.
  - Ao final, as tuplas resultantes são reunidas para obter o resultado.
  - Caso especial: Junção assimétrica, onde *n=1*.

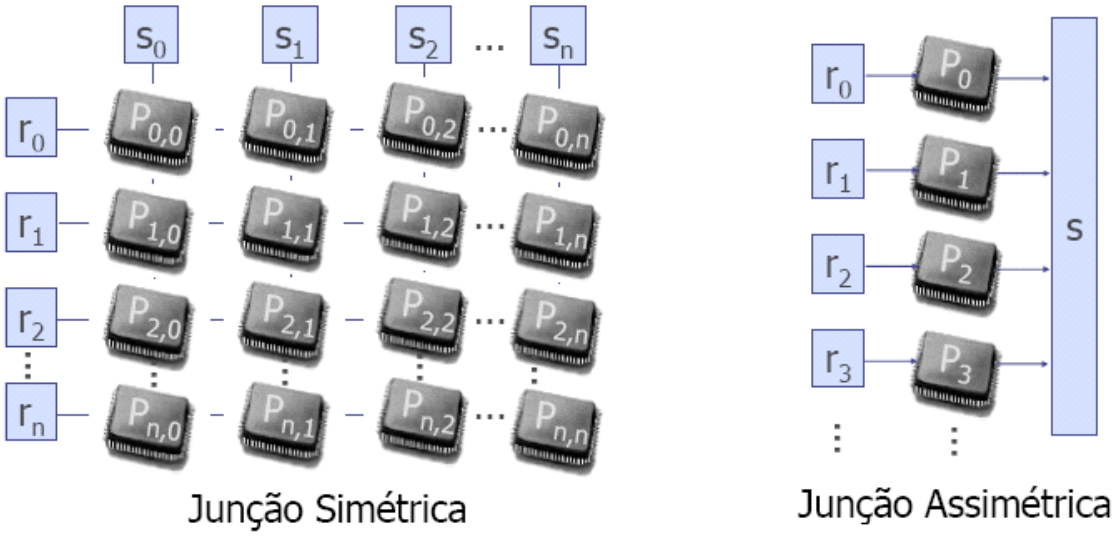


Figura 11: Algoritmo de junção por fragmentação e replicação

- **Algoritmo de junção paralela de laço aninhado:** É utilizado quando uma das relações é muito menor do que a outra.

  – Cada processador faz a junção indexada de laço aninhado da relação menor, replicada em todos os processadores, com uma partição da relação maior.

  – Utiliza índice da relação maior no atributo de junção.

Capítulo 4

Projetos dos SBDPs

- Paralelização no armazenamento de dados
- Paralelização no processamento de consultas
- Carregamento paralelo de dados a partir de fontes externas
- Resistência à falha de alguns processadores ou discos
- Reorganização on-line de dados e troca de esquemas

Capítulo 5

Conclusões

Neste trabalho foi apresentado processamento paralelo aplicado ao Sistema de Banco de Dados. Todas as aplicações modernas, tais como automação de escritórios, sistemas de apoio à decisão, sistemas de trabalho cooperativo, inteligência artificial distribuída, dependem, de alguma forma, da eficiência no acesso à informação. Sistemas Paralelos representam uma técnica que possibilita agilizar este desempenho de acesso.

A natureza de muitas aplicações e a necessidade de maior desempenho no processamento destas aplicações levaram ao surgimento dos bancos de dados paralelos. Na busca pelo alto desempenho tem-se procurado evitar tráfego excessivo de dados, para não prejudicar o tempo de resposta dessas aplicações.

Descrevemos as técnicas de fragmentação e alocação de dados, de processamento e otimização de consultas em um banco de dados paralelo, os tipos de paralelismos, bem como detalhes de arquitetura. Sistemas de banco de dados facilitam a exploração de inúmeros hardwares interligados em uma ou várias máquinas, compartilhando, assim, os dados e processamento entre eles.

Este trabalho foi de grande importância para compreender os conceitos, histórico e funcionamento dos Sistemas de Banco de Dados Paralelos.

Apêndice

Questões relacionadas a SBDPs

a) Quais são as quatro categorias em que os SBDPs podem ser classificados? Explique o funcionamento de cada uma delas.

- *Memória compartilhada:* Todos os processadores compartilham a mesma memória e os mesmos discos, geralmente por meio de cabo ou de rede de interconexão.
- *Discos compartilhados:* Todos os processadores podem acessar diretamente os discos por meio de uma rede de conexão, mas cada um deles possui uma memória exclusiva.
- *Sem compartilhamento:* Cada equipamento de um nó consiste em um processador, uma memória e discos.
- *Hierárquico:* Cada nó pode ser considerado um sistema independente.

b) Quais são as três formas de utilização do paralelismo nos sistemas gerenciadores de banco de dados?

- Parale
- llismo na entrada e saída de dados
- Paralelismo no processamento de consultas
- Paralelismo no processamento de operações

c) Quais são os três tipos de particionamento de dados para obtenção de paralelismo de entrada e saída de dados e como ocorre a distribuição das tuplas?

- *Particionamento horizontal:* As tuplas de uma relação são divididas entre muitos discos, e cada tupla reside em um disco diferente.
- *Particionamento vertical:* Os campos de uma relação se particionam entre os discos, onde cada campo reside em um ou mais discos.
- *Particionamento misto:* É obtido por meio da combinação entre os particionamentos horizontal e vertical, ou o contrário.

Referências Bibliográficas

http://www.cos.ufrj.br/~marta/papers/TeseMauroS.pdf

http://www.shammas.eng.br/acad/sitesalunos0606/bdpar/sis_arq.html

http://www.jsoares.net/CEFET/BD/sbdp.pdf

ftp://ftp.dca.fee.unicamp.br/pub/docs/ricarte/apostilas/spbdaa.pdf

http://64.233.269.104/search?q=cache:S1Dt6glMChoJ:www.cos.ufrj.bt/~marta/tuttext.ps+banco+de+dados+paralelos&hl=pt-BR&ct=clnk&cd=5&gl=br

FORNARI, Miguel Rodrigues. **Sistemas Gerenciadores de Bancos de Dados Geográficos Distribuídos e Paralelos**. – Porto Alegre: PGCC da UFRGS, 2002.

SILBERSCHATZ, A.; KORTH, H.F. SUDARSHAN, **S. Sistemas de Bancos de Dados**, 3ª ed. São Paulo: Makron Books. 1999. 779 pp.