UNIVERSIDADE DE BRASÍLIA

Faculdade do Gama

Sistemas de Banco de Dados 2

Tecnologias de Banco de Dados (TI-BD)

Bancos de Dados Paralelos

Guilherme Guy de Andrade - 16/0123186

Brasília, DF

2019

Definição da Tecnologia

Com o avanço da tecnologia de processadores de múltiplos núcleos alguns Sistemas Gerenciadores de Bancos de Dados (SGDB) procuram se aproveitar desta novidade ao operar seus Bancos de Dados (BD) de forma a obter maior desempenho com um menor custo, já que se pode fazer mais com menos máquinas (VALDURIEZ, 1993).

Isso implica na utilização de técnicas de processamento paralelo de dados ao realizar operações no BD. Por definição, paralelismo em computação é a execução de duas ou mais instruções ao mesmo tempo. A quantidade de instruções executadas depende da quantidade de núcleos que um processador tem à sua disposição.

O objetivo de utilizar paralelismo em computação é poder aproveitar o poder de processamento de vários núcleos, trabalhando de forma cooperativa, em uma aplicação, seja para dividir o peso de uma tarefa específica ou realizar tarefas independentes ao mesmo tempo, dando maior rapidez ao programa de computador em execução.

Em um BD a utilização de paralelismo significa, entre outras aplicações, dividir o trabalho necessário para realizar alguma tarefa que o SGDB deseja executar ou aceitar que múltiplas tarefas sejam realizadas ao mesmo tempo, de forma sincronizada, sem interferir uma com a outra (SILBERSCHATZ, 2016).

Um exemplo é a execução de duas instruções de busca de dados de forma simultânea. Observe que este exemplo se encaixa na categoria de tarefas que não possuem dependência entre si, ou seja, não faz diferença realizar a tarefa A e depois a tarefa B ou a tarefa B e depois a tarefa A.

Em alguns casos a ordem das tarefas tem influência no resultado final, imagine que a tarefa A seja uma tarefa de escrita de dados e a tarefa B seja de busca de dados, ambas disparadas ao mesmo tempo. Uma das dificuldades existentes com a tecnologia de processamento paralelo é saber a ordem em que estas tarefas devem ser executadas e garantir a sincronização das tarefas em andamentos.

Uma outra forma de se acelerar o acesso à um BD é por meio de *hardware* específico. Entretanto isso acrescenta custos relacionados ao aspecto físico (*hardware*) da máquina em que o BD será operado. Por este motivo, as aplicações de paralelismo vem sendo realizadas muito mais em nível de *software* aproveitando o *hardware* mais genérico e mais barato que se desenvolveu ao longo dos anos, e não feito sob medida.

Uma associação comum aos SGDBs que utilizam paralelismo é a tecnologia de BDs distribuídos. Muitas vezes os termos são utilizados como sinônimos, inclusive existem dificuldades de se concluir onde exatamente as duas tecnologias se separam (ÖZSU, 1996). Existem sistemas que são paralelos e distribuídos ao mesmo tempo.

Um SGDB distribuído, superficialmente, é um SGBD capaz de operar em múltiplas máquinas ao mesmo tempo. Já as operações de paralelismo são

mais voltadas a executar múltiplas tarefas no processador de uma máquina que esteja executando um SGDB.

A arquitetura de um SGDB paralelo pode assumir diversas formas, desde arquiteturas *shared-noithing* (sem compartilhamento) até arquiteturas *shared-memory* (memória compartilhada). Quanto mais elementos são compartilhados pelos núcleos dos processadores da máquina que abriga o SGDB, mais próximo ele se torna da arquitetura *shared-memory* e quanto menos compartilhamento mais próximo se torna da arquitetura *shared-nothing*. É comum se utilizar uma arquitetura *shared-disk* como ponto intermediário entre os dois extremos arquiteturais (ÖZSU, 1996).

Em uma arquitetura *shared-nothing* nada é compartilhado e cada processador tem sua própria memória volátil e seus próprios discos de armazenamento permanente de dados.

Já em uma arquitetura *shared-memory* cada processador pode acessar qualquer parte da memória e tem acesso a qualquer disco de armazenamento permanente.

E, por fim, na arquitetura *shared-disk* os processadores tem sua própria memória volátil mas compartilham discos de armazenamento permanente de dados.

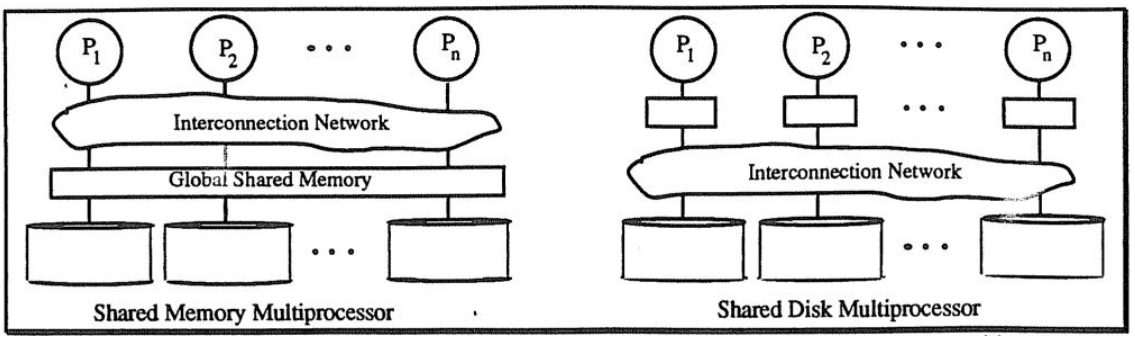

Figura 1: Demonstração da região de interconexão entre processadores em um SGDB paralelo. Fonte: DEWITT, 1992.

Na figura 1 pode ser observada a partir de que ponto os processadores se conectam em arquiteturas *shared-memory* e *shared-disk*. Observe que na arquitetura *shared-disk* a memória se torna individual para cada processador, o que pode trazer vantagens relacionadas à segurança e confiabilidade do sistema, quando comparada com a arquitetura *shared-memory*, já que a chance de se corromper a memória é menor, pois há apenas um processador à acessando.

A forma de organizar os dados também também recebe influência da arquitetura. Segundo Dewitt em “Parallel database systems: The future of database processing or a passing fad?” na arquitetura *shared-nothing* as tuplas de uma relação são salvas em discos separados, sendo responsabilidade de cada processador recuperar a tupla e enviar para outro processador por meio de redes de informação. Observe que a arquitetura *shared-nothing* se assemelha muito a um SGDB distribuído (ÖZSU, 1996).

Silberschatz em “Sistema de banco de dados” lista três formas de organizar os dados em um SGDB paralelo.

A primeira técnica se chama rodízio (também conhecido como *round-robin* em inglês). Ela consiste em fazer o rodízio do disco em que se vai escrever um dado. Por exemplo: se existirem três discos nomeados *d1*, *d2* e *d3* em um sistema de banco de dados e se quer escrever seis tuplas nomeadas *t1*, *t2* … *t5* e *t6* se escreveria *t1* em *d1*, *t2* em *d2*, *t3* em *d3*, *t4* em *d1*, *t5* em *d2* e *t6* em *d3*. Esse método é recomendado quando se possui muitas leituras sequenciais de dados, pois dividirá o tempo necessário para leitura em vários discos, mas dificulda leituras de acesso direto a um dado ou leituras em um intervalo específico.

A segunda técnica é o particionamento por *hash*. Onde é gerado um *hash* para algum atributo das tuplas de uma relação e se tem os dados organizados seguindo a ordem deste *hash*. Dessa forma se pode descobrir exatamente em que disco buscar uma tupla calculando o *hash* do dado inserido como entrada na busca, direcionando os esforços de leitura somente ao disco em que se sabe que o dado está localizado. Entretanto, o particionamento por *hash* não tornará mais rápida uma busca que dependa de um atributo que não tenha um *hash* indexado no banco de dados.

A terceira técnica é o particionamento por sequência. É escolhido um atributo e a partir dele os dados são separados nos discos com base nos limites. Se existem dois discos nomeados *d1* e *d2* e, além disso, um atributo de uma relação chamado *x* cujo valor varie de zero até mil, pode-se definir um esquema de particionamento por sequência que salve tuplas com *x* menor que quinhentos em *d1* e valores maiores ou iguais a quinhentos em *d2*. Esta técnica pode deixar os dados fragmentados entre os discos, de forma que alguns tenham poucos dados e outros muitos dados. Em discos com poucos dados a busca será rápida, enquanto em outros a velocidade pode ser consideravelmente reduzida.

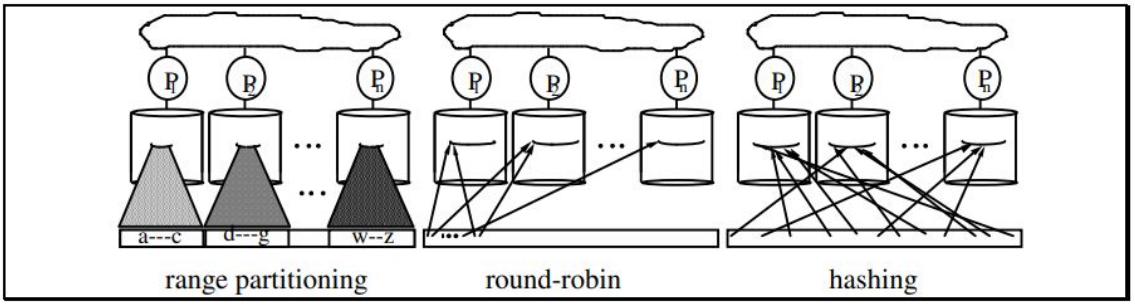

Figura 2: Exibição gráfica das técnicas de particionamento de dados. Da esquerda para direita: particionamento por sequência, por rodízio e por hash. Fonte: DEWITT, 1990.

Objetivos da Tecnologia

A aplicação de paralelismo em SGDBs tem dois objetivos principais: aumento de desempenho e avaliabilidade em SGDBs e diminuição de custos

para manter e criar a máquina hospedeira do SGDB.

Foi constatada a dificuldade de se construir máquinas potentes o suficiente para servir milhares de operações de leitura e escrita de dados em bases de dados de tamanhos significativamente grandes, enquanto processadores de múltiplos núcleos vinham se tornando cada vez mais potentes e financeiramente acessíveis (DEWITT, 1992).

Outra vantagem é que as máquinas hospedeiras de SGDBs poderiam passar a ser construídas com componentes comerciais e não mais necessitam de peças feitas especificamente para elas.

Ainda de acordo com Dewitt, em relação a velocidade os SGDBs poderiam contar com aumento linear de velocidade, ou seja, o dobro de *hardware* pode executar uma tarefa na metade do tempo e também aumento linear em escala: o dobro de *hardware* pode executar uma tarefa com o dobro de peso/dificuldade no mesmo tempo. Nem sempre estes aumentos são constantes, dependendo da arquitetura e outros fatores, sendo necessário realizar testes e comparações para comprovar se o sistema em execução atende à estas regras.

Geralmente a utilização do paralelismo por SGBDs tende a promover um aumento de velocidade e de avaliabilidade das informações guardadas no BD. E por isso tem se tornado uma tecnologia popular para BDs.

Vantagens da Tecnologia Pesquisada

A tecnologia de paralelismo em BDs teve como fator que apoiou sua popularização o crescimento de bancos de dados relacionais (DEWITT, 1992). Dessa forma os bancos de dados paralelos tem forte conexão com bancos de dados relacionais. Um exemplo disso é a possibilidade de salvar tuplas de relações em discos diferentes e possibilitar seu acesso de forma independente. Assim o SGDB poderia atender a uma solicitação de escrita e uma de leitura ao mesmo tempo, desde que não interferissem uma na outra.

Em BDs, por causa de sua natureza ordenada é possível separar os dados em discos diferentes. Em um BD relacional isto é particularmente interessante pois se torna possível salvar algumas tuplas de uma relação em um disco e outras tuplas em outro disco, de acordo com critérios predefinidos, o que pode acelerar a velocidade de buscas de dados pois os discos podem ser acessados de forma independente e ao mesmo tempo, conforme técnicas de particionamento já discutidas (SILBERSCHATZ, 2016).

O paralelismo é uma forma economicamente mais eficiente de se aumentar a velocidade e a escala de acesso aos dados em um BD. Além disso, por causa do crescimento da *internet* e da necessidade de se acessar informações ao mesmo tempo (por causa de requisições feitas por usuário) o BD paralelo se torna relevante, pois oferece ferramentas adequadas para lidar com os problemas relacionados a este contexto, diminuindo o investimento financeiro necessário para criar uma infraestrutura que possa servir os usuários

finais que tentam acessar algum serviço ou plataforma em tempo hábil.

Além disso, BDs paralelos oferecem ferramentas para lidar com grandes quantidades de dados, na ordem dos *terabytes* (SILBERSCHATZ, 2016), de forma muito mais rápida que um BD sequencial. Um exemplo é a utilização de BDs paralelos em pesquisas científicas, onde se pode processar uma quantidade maior de dados, o que aumenta a precisão dos resultados, além de poder resolver problemas que seriam impossíveis em um BD sequencial (VALDURIEZ, 1993).

Observe que estes todos estes pontos vantajosos podem ser executados em um BD relacional. O paralelismo é uma tecnologia aplicada a nível de SGBD. Portanto, se o SGDB suportar um paradigma de estrutura de BD, o paralelismo funcionará neste paradigma.

Desvantagens da Tecnologia Pesquisada

Por mais que se tenham vantagens na fragmentação dos dados, isso também oferece desafios quando se utiliza a tecnologia de paralelismo em BDs. O posicionamento dos dados é essencial para que os processadores não fiquem sobrecarregados. Se há uma distribuição não uniforme de dados o paralelismo pode levar a desempenho pior do que o processamento em único processador (VALDURIEZ, 1993).

Uma técnica utilizada para garantir a distribuição de dados é a referenciada pelo termo em inglês *declustering*. Esta técnica propõe a distribuição horizontal das tuplas de uma relação em diferentes discos, de acordo com técnicas discutidas anteriormente como particionamento por *hash* ou intervalo ou particionamento de rodízio, cada método de particionamento com suas vantagens e desvantagens.

Outra forma de combater a distribuição desigual de dados nos discos de um SGDB é a reorganização dinâmica de dados (VALDURIEZ, 1993).

Além disso, garantir a consistência dos dados pode se tornar um problema, pois quando se utiliza um sistema computacional com paralelismo se encontram problemas como condições de corrida e *deadlocks*. Ao se preparar uma arquitetura paralela deve ser tomado cuidado para que estes fenômenos possam ser evitados. Isso contribui para que o código de um SGDB com suporte a paralelismo não seja trivial e, por consequência, com dificuldade de manutenção mais elevada.

Uma preocupação relacionada é a alta disponibilidade de dados para sistemas de grande porte. É prática comum se ter ao menos duas versões de um mesmo arquivo (original e cópia de *backup*). Entretanto em um sistema de grande porte isso pode se tornar um problema por causa da fragmentação de dados, onde uma falha pode destruir dados e impossibilitar o sistema como um todo de continuar operando para não perder um estado de coerência nos dados (VALDURIEZ, 1993).

Existem alternativas que buscam solucionar isto, como a duplicação de

dados não só para *backup* mas também para oferta de dados em um esquema de *hot-swap*, onde se o dado original estava na fonte *d1* e replicado em *d2* (além de seus *backups*) e a fonte *d1* sofrer uma falha catastrófica, o dado poderá continuar a ser acessado em *d2* até que se realize um reparo em *d1*. Esta técnica também é muito utilizada em SGDBs distribuídos.

Em relação a velocidade, nem todas as operações que se realizam em BDs podem ser beneficiadas pelo paralelismo, inclusive existindo operações que são prejudicadas pelo uso desta tecnologia. Operações que precisam consumir todos os dados de entrada antes de produzir sua saída, como operações que necessitam ordenar os dados, são um exemplo disso (DEWITT, 1990).

Assim como as vantagens, as desvantagens de um SGDB paralelo não se atém a um paradigma específico de BD. Principalmente por ser uma tecnologia geralmente implementada em nível de SGDB.

Exemplos de Uso

Geralmente a utilização de um SGDB com suporte a paralelismo está relacionada ao uso da tecnologia de BDs distribuídos. Um exemplo comercial é a implementação *MySQL Cluster* que é um híbrido da tecnologia paralela com a tecnologia distribuída.

Outra implementação de SGDB com suporte a paralelismo é o MongoDB. Além de paralelo, é um SGDB distribuído e não relacional. Também é um software *open-source* de licença *Server Side Public License* (SSPL), que é baseada na *GNU General Public License 3* (GPL3).

O MongoDB utiliza paralelismo em suas operações de acesso ao BD, seja ele apenas em um único nó ou em diversos nós. Também aplicam a técnica de *sharding*, espalhando suas coleções (semelhantes a relações em BDs relacionais) em diversos nós.

Estas implementações de SGDBs com tecnologia paralela são amplamente utilizadas, com vários histórias de sucesso documentadas.

Uma delas é descrita por Mat Keep em “Cisco & MongoDB: E-commerce Transformation”. Onde é descrito o relato de Dharmesh Panchmatia, Diretor de *e-commerce* na *Cisco Systems*. Segundo o relato, a utilização do SGDB MongoDB possibilitou a Cisco migrar sua plataforma de *e-commerce* para uma plataforma em nuvem e melhorar a experiência do usuário final com tempos de resposta até oito vezes mais rápidos que os encontrados anteriormente e com disponibilidade de dados constante.

Um fator decisivo para a Cisco escolher este tipo de tecnologia foi a sua capacidade de expandir para atender cargas de trabalho maiores e também de resistir a falhas catastróficas em diversos pontos da rede formada por seus nós.

É possível observar que a diminuição do tempo médio de resposta à solicitações ao BD também tem influência no usuário final, que precisará esperar menos tempo para continuar com a utilização do sistema.

Bibliografía

DEWITT, David J.. ; GRAY, Jim. Parallel database systems: The future of database processing or a passing fad?. **SIGMOD record**, v. 19, n. 4, p. 104-112, 1990.

DEWITT, David J.; GRAY, Jim. **Parallel database systems: The future of high performance database processing**. University of Wisconsin-Madison Department of Computer Sciences, 1992.

KEEP, Mat. Cisco & MongoDB: E-commerce Transformation. **Cisco & MongoDB: E-commerce Transformation**. [*S. l.*], 30 nov. 2016. Disponível em: https://www.mongodb.com/blog/post/cisco-and-mongodb-e-commerce-transformation. Acesso em: 6 set. 2019.

MONGODB, INC. **FAQ: Concurrency**. [*S. l.*], 201-. Disponível em: https://docs.mongodb.com/manual/faq/concurrency/. Acesso em: 6 set. 2019.

MONGODB, INC. **Server Side Public License FAQ**. [*S. l.*], 201-. Disponível em: https://www.mongodb.com/licensing/server-side-public-license/faq. Acesso em: 6 set. 2019.

MONGODB, INC. **Our Customers**. [*S. l.*], 201-. Disponível em: https://www.mongodb.com/who-uses-mongodb. Acesso em: 6 set. 2019.

ORACLE CORPORATION. **MySQL Cluster CGE**. [S. l.], 201-. Disponível em: https://www.mysql.com/products/cluster/. Acesso em: 6 set. 2019.

ÖZSU, M. Tamer; VALDURIEZ, Patrick. Distributed and parallel database systems. **ACM Computing Surveys**, v. 28, n. 1, p. 125-128, 1996.

SILBERSCHATZ, Abraham; SUNDARSHAN, S.; KORTH, Henry F. **Sistema de banco de dados**. Elsevier Brasil, 2016.

VALDURIEZ, Patrick. Parallel database systems: Open problems and new issues. **Distributed and parallel Databases**, v. 1, n. 2, p. 137-165, 1993.