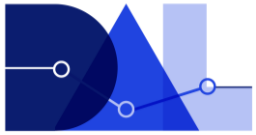
CEFET/RJ



BANCO DE DADOS PARALELO E DISTRIBUÍDO

Eduardo Ogasawara
eogasawara@ieee.org
https://eic.cefet-rj.br/~eogasawara

Arquiteturas de Banco de Dados

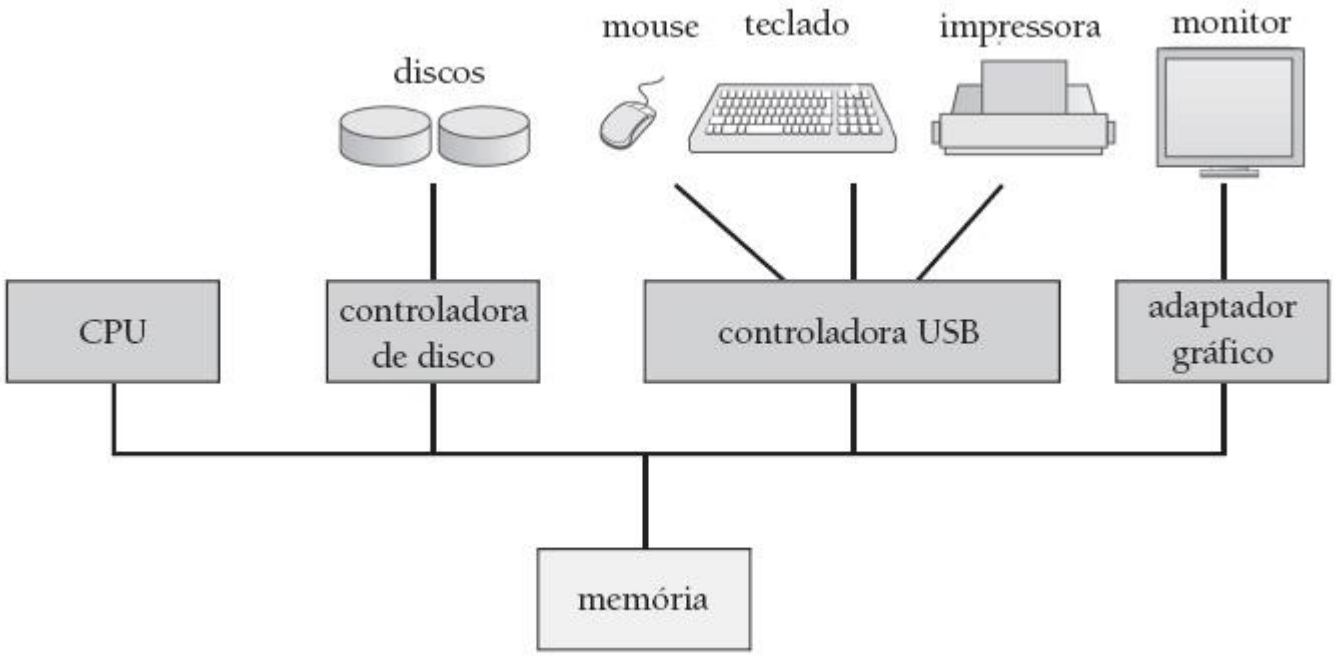
Arquiteturas de sistema de banco de dados

- Sistemas centralizados
- Sistemas cliente-servidor
- Sistemas paralelos
- Sistemas distribuídos
- Tipos de rede

Sistemas centralizados

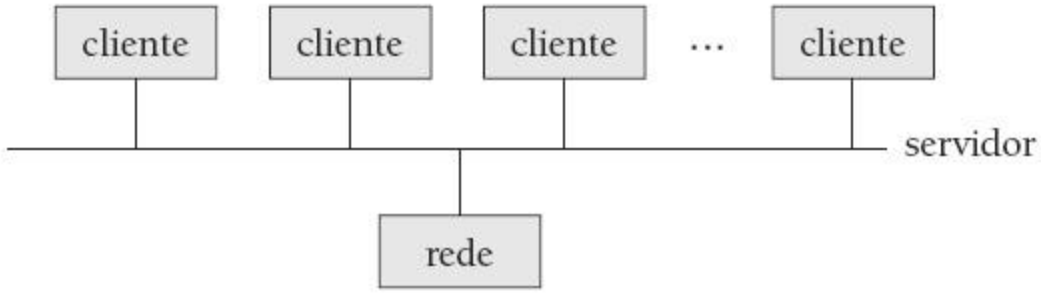
- Executados em um único computador e não interagem com outros sistemas de computador
- Sistema de computador de uso geral
  - uma ou mais CPUs e uma série de controladoras de dispositivo que estão conectadas por meio de um barramento comum que oferece acesso à memória compartilhada
- Sistema de único usuário
- Sistema multiusuário
  - Várias CPUs e SO multiusuário
  - Atende a diversos usuários que estão conectados ao sistema por terminais. Normalmente chamados de sistemas servidores

Um sistema de computador centralizado



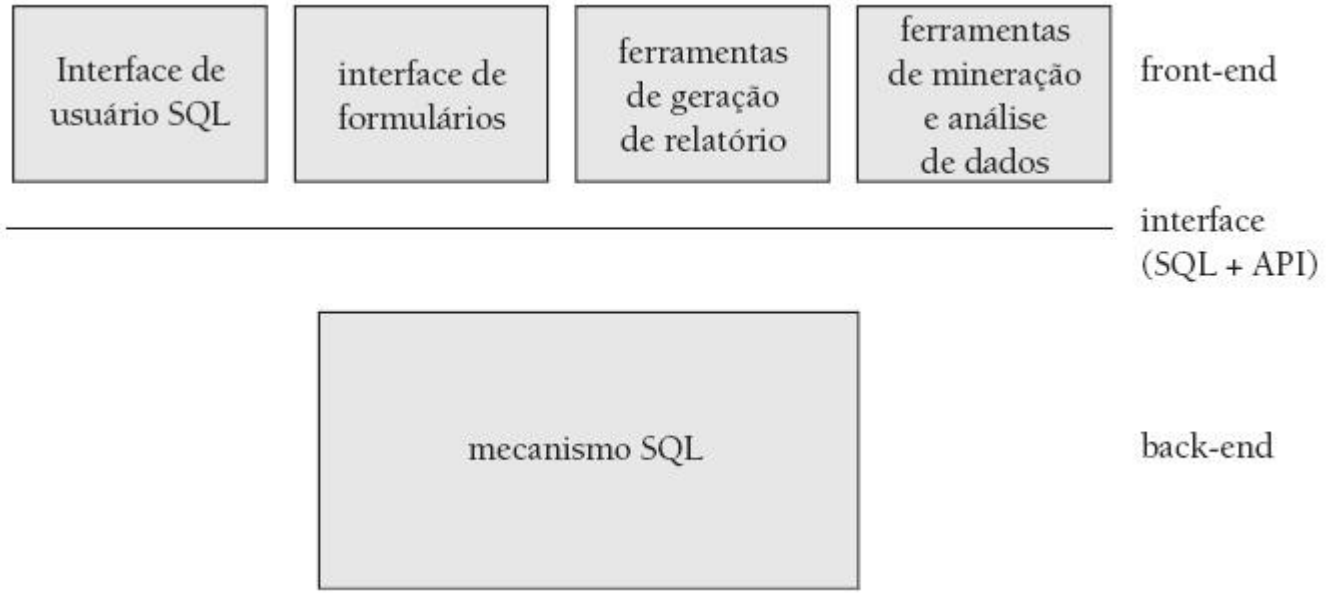
Sistemas cliente-servidor

- Sistemas servidores satisfazem solicitações geradas em m sistemas cliente, cuja estrutura geral aparece a seguir:



Sistemas cliente-servidor (cont.)

- A funcionalidade do banco de dados pode ser dividida em:
  - Back-end: controla as estruturas de acesso, avaliação e otimização de consulta, controle de concorrência e recuperação
  - Front-end: consiste em ferramentas como formulários, criadores de relatório e facilidades de interface gráfica com o usuário
- A interface entre o front-end e o back-end é por meio da SQL ou por uma interface de programa de aplicação



Sistemas cliente-servidor (cont.)

- Vantagens de substituir mainframes por redes de estações de trabalho ou computadores pessoais conectados a máquinas servidoras de back-end:
  - funcionalidade melhor para o custo
  - flexibilidade na localização de recursos e facilidades de expansão
  - melhores interfaces com o usuário
  - manutenção mais fácil
- Os sistemas servidores podem ser amplamente categorizados em dois tipos:
  - servidores de transação: muito usados em sistemas OLTP
  - servidores de dados: muito usados em sistemas OLAP

Servidores de transação

- Também chamados sistemas servidores de consulta ou sistemas servidores SQL
  - clientes enviam solicitações ao sistema servidor onde as transações são executadas, e os resultados são entregues de volta ao cliente
  - Solicitações especificadas em SQL e comunicadas ao servidor por meio de um mecanismo de Remote Procedure Call (RPC)
- A RPC transacional permite que muitas chamadas de RPC formem coletivamente uma transação
  - Open Database Connectivity (ODBC) é uma interface de programa de aplicação em linguagem C padrão da Microsoft para conectar a um servidor, enviando solicitações SQL e recebendo resultados
  - O padrão JDBC semelhante a ODBC, para Java

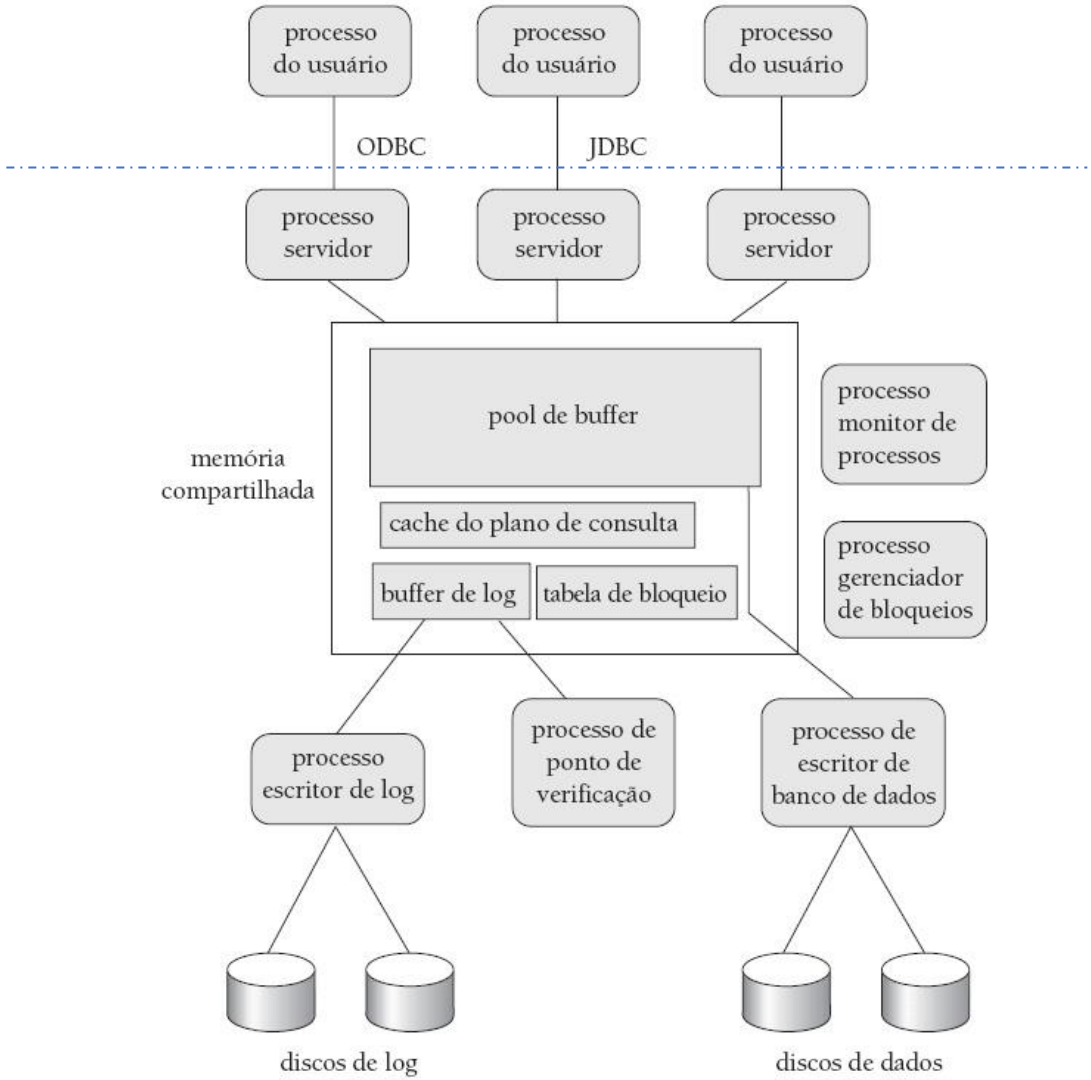
Estrutura do processo servidor de transação

- Um servidor de transação típico consiste em vários processos acessando dados na memória compartilhada
- Processos servidores
  - Estes recebem consultas do usuário (transações), executam-nas e enviam resultados de volta
  - Os processos podem ser multithreaded, permitindo que um único processo execute várias consultas do usuário simultaneamente
  - Normalmente, vários processos servidores multithreaded
- Processo gerenciador de bloqueio
- Processo escritor de banco de dados
  - Envia blocos de buffer modificados continuamente para os discos

Processos servidores de transação (cont.)

- Processo escritor de log
  - Processos servidores simplificam a inclusão de registros de log no buffer de registro de log
  - Processo escritor de log envia registros de log ao armazenamento estável
- Processo de ponto de verificação
  - Realiza pontos de verificação periódicos
- Processo monitor de processos
  - Monitora outros processos e toma ações de recuperação se algum dos outros processos falhar
    - Por exemplo, abortar quaisquer transações sendo executadas por um processo servidor e reiniciá-lo

Processos do sistema de transação (cont.)



Processos do sistema de transação (cont.)

- Memória compartilhada contém dados compartilhados
  - Pool de buffer
  - Tabela de bloqueio
  - Buffer de log
  - Planos de consulta em cache (reutilizados se a mesma consulta for submetida novamente)
- Todos os processos de banco de dados podem acessar a memória compartilhada
- Para garantir que dois processos não estão acessando a mesma estrutura de dados ao mesmo tempo, os sistemas de banco de dados implementam a exclusão mútua usando
  - Semáforos do sistema operacional
  - Instruções indivisíveis como testar-e-definir

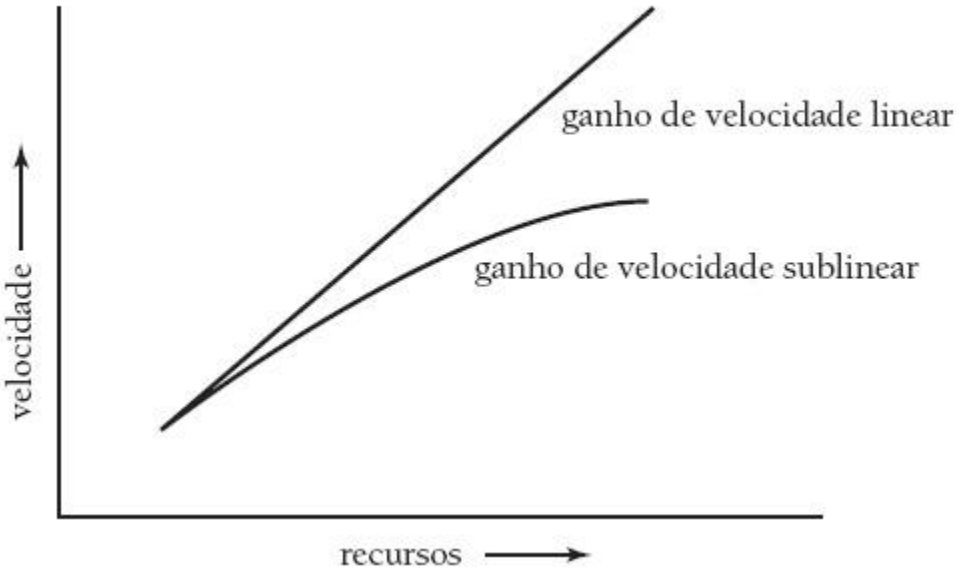
Sistemas paralelos

- Os sistemas de banco de dados paralelos consistem em vários processadores e vários discos conectados por uma rede de interconexão rápida
- Uma máquina paralela grão grosso (do inglês, coarse-grain) consiste em um pequeno número de processadores poderosos
- Uma máquina maciçamente paralela ou paralela grão fino (do inglês, fine grain) utiliza milhares de processadores menores
- Duas medidas de desempenho principais:
  - Vazão (throughput) - o número de tarefas que podem ser completadas em determinado intervalo de tempo
  - tempo de resposta - o tempo gasto para completar uma única tarefa desde o momento em que ela é submetida

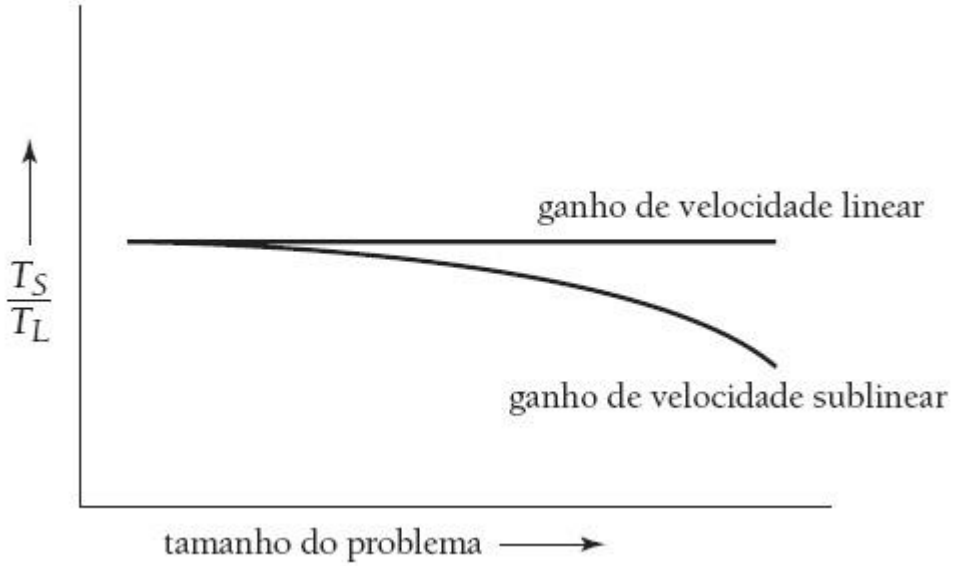
Ganho de velocidade e ganho de escala

- Ganho de velocidade: um problema de tamanho fixo executando em um sistema pequeno é dado a um sistema que é N vezes maior.
  - Medido por:
  - $$\textbf{\textit{ganho de velocidade}} = \frac{\text{tempo gasto no sistema pequeno}}{\text{tempo gasto no sistema grande}}$$
  - O ganho de velocidade é linear se a equação for igual a N.
- Ganho de escala: aumenta o tamanho do problema e do sistema
  - Sistema N vezes maior usado para realizar tarefa N vezes maior
  - Medido por:
  - $$\text{ganho de escala} = \frac{\text{tempo gasto com problema pequeno no sistema pequeno}}{\text{tempo gasto com problema grande no sistema grande}}$$
  - O ganho de escala é linear se a equação for igual a 1

Ganho de velocidade



Ganho de escala



Ganho de escala em batch e transação

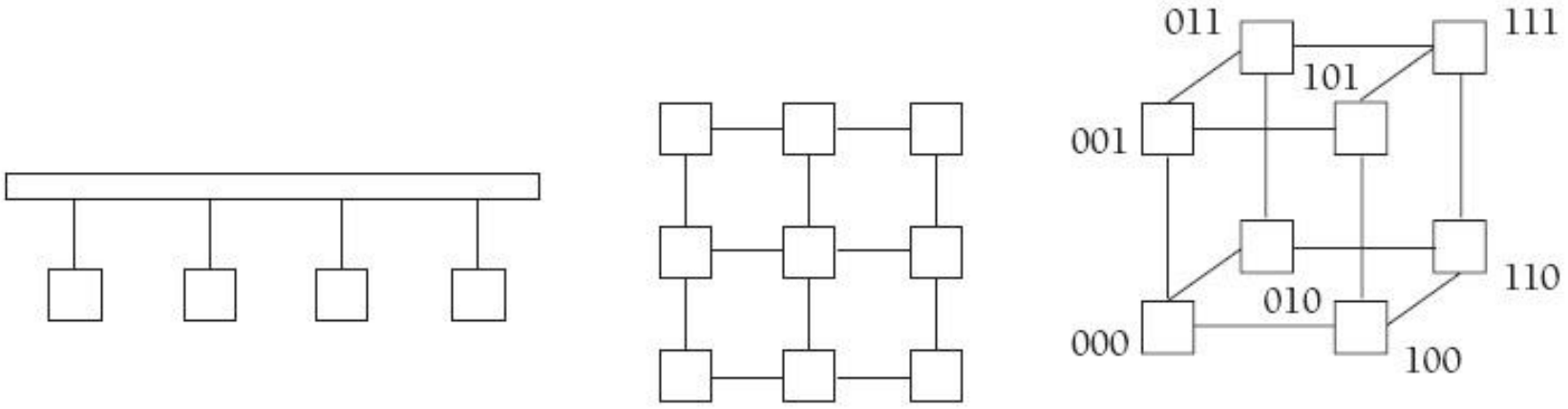
- Ganho de escala de batch:
  - Uma única tarefa grande; tópica da maioria das consultas de banco de dados e simulação científica
  - Use um computador N vezes maior no problema N vezes maior
- Ganho de escala de transação:
  - Diversas consultas pequenas submetidas por usuários independentes a um banco de dados compartilhado; sistemas típicos de processamento de transação e tempo compartilhado
  - N vezes mais usuários submetendo solicitações (logo, N vezes mais solicitações) a um banco de dados N vezes maior, em um computador N vezes maior
  - Bem adequado para a execução paralela

Fatores limitando o ganho de velocidade e ganho de escala

- Ganho de velocidade e ganho de escala normalmente são sublineares devido a:
  - Custos de partida: O custo para iniciar vários processos pode dominar o tempo da computação, se o grau de paralelismo for alto
  - Interferência: Os processos acessando os recursos compartilhados (por exemplo, barramento do sistema, discos ou bloqueios) competem entre si, gastando tempo para esperar por outros processos, em vez de realizar um trabalho útil
  - Distorção: Aumentar o grau de paralelismo aumenta a variância nos tempos de serviço das tarefas executando em paralelo. O tempo de execução geral determinado pela mais lenta das tarefas executando em paralelo

Arquiteturas de rede de interconexão

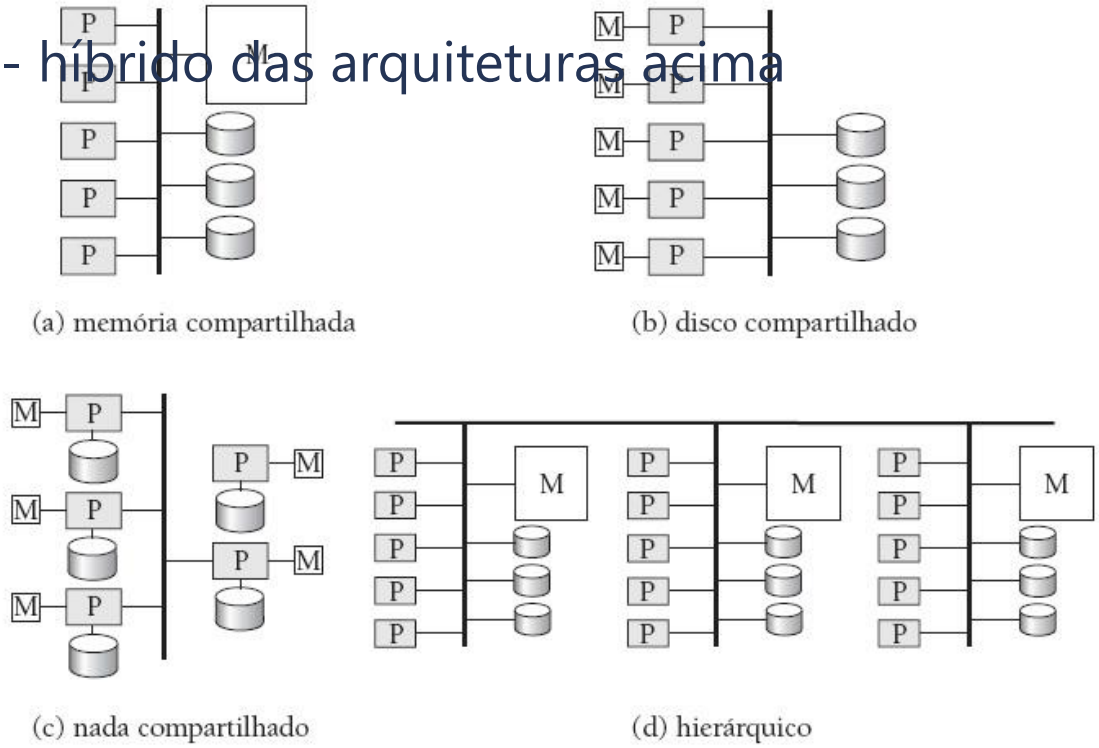
- Barramento. Componentes do sistema enviam dados e recebem dados de um único barramento de comunicação
- Malha. Componentes são arrumados como nós em uma grade, e cada componente é conectado a todos os componentes adjacentes
- Hipercubo. Componentes são numerados em binário; componentes são conectados entre si se suas representações binárias diferirem exatamente em um bit



(a) barramento (b) malha (c) hipercubo

Arquiteturas de bancos de dados paralelas

- Memória compartilhada: processadores compartilham memória comum
- Disco compartilhado - processadores compartilham um disco comum
- Nada compartilhado - processadores não compartilham memória e disco
- Hierárquica - híbrido das arquiteturas acima



Memória compartilhada

- Processadores e discos têm acesso a uma memória comum, normalmente por meio de um barramento ou por uma rede de interconexão
- Comunicação extremamente eficiente entre os processadores - os dados na memória compartilhada podem ser acessados por qualquer processador sem ter que movê-los usando software
- Desvantagem - arquitetura não é expansível além de 32 ou 64 processadores, pois o barramento ou a rede de interconexão se torna um gargalo
- Muito usada para graus de paralelismo menores (4 a 8)

Disco compartilhado

- Todos os processadores podem acessar diretamente todos os discos por meio de uma rede de interconexão, mas os processadores têm memórias privadas
  - O barramento da memória não é um gargalo
  - A arquitetura oferece um grau de tolerância a falhas - se um processador falhar, os outros processadores podem assumir suas tarefas, pois o banco de dados é residente nos discos que são acessíveis por todos os processadores
- Desvantagem: gargalo agora ocorre na interconexão com o subsistema de disco
- Sistemas de disco compartilhado podem se expandir até um número um pouco maior de processadores, mas a comunicação entre os processadores é mais lenta

Nada compartilhado

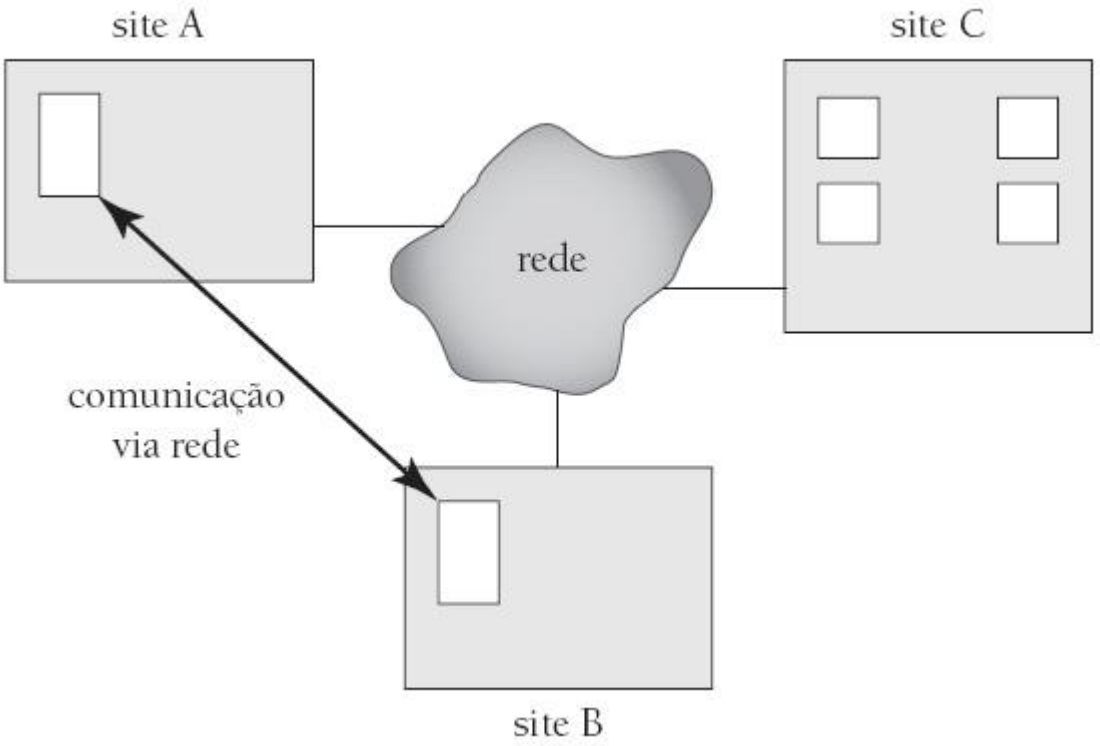
- O nó consiste em um processador, memória e um ou mais discos
- Os processadores em um nó se comunicam com outro processador em outro nó, usando uma rede de interconexão
- Um nó funciona como servidor para os dados no disco ou discos que o nó possui
- Os dados acessados a partir de discos locais (e acessos a memória locais) não passam pela rede de interconexão, minimizando assim a interferência do compartilhamento de recursos
- Vantagem: Os multiprocessadores de nada compartilhado podem ser expandidos até milhares de processadores sem interferência
- Desvantagem: custo de comunicação e acesso a disco não local, o envio de dados envolver a interação do software nas duas pontas

Hierárquica

- Combina características de arquiteturas de memória compartilhada, disco compartilhado e nada compartilhado
- Nível superior é uma arquitetura nada compartilhado - os nós conectados por uma rede de interconexão e não compartilha discos ou memória entre si
- Cada nó do sistema poderia ser um sistema de memória compartilhada com alguns processadores
- Como alternativa, cada nó poderia ser um sistema de disco compartilhado, e cada um dos sistemas compartilhando um conjunto de discos poderia ser um sistema de memória compartilhada
- Reduz a complexidade de programar tais sistemas por arquiteturas de memória virtual distribuída
  - Também chamada arquitetura de memória não uniforme

Sistemas distribuídos

- Dados espalhados por várias máquinas (também chamadas de sites ou nós)
- Rede interliga as máquinas
- Dados compartilhados por usuários em várias máquinas



Bancos de dados distribuídos

- Bancos de dados distribuídos homogêneos
  - Mesmo software/esquema em todos os sites, os dados podem estar particionados entre os sites
  - Objetivo: oferecer uma visão de um único banco de dados, ocultando os detalhes da distribuição
- Bancos de dados distribuídos heterogêneos
  - Software/esquema diferente em sites diferentes
  - Objetivo: integrar bancos de dados existentes para oferecer funcionalidade útil
- Diferenciar entre transações locais e globais
  - Uma transação local acessa dados no único site em que a transação foi iniciada
  - Uma transação global acessa dados em um site diferente daquele em que a transação foi iniciada ou acessa dados em vários sites diferentes

Escolhas nos sistemas distribuídos

- Compartilhamento de dados
  - usuários em um site capazes de acessar os dados residindo em alguns outros sites
- Autonomia
  - cada site é capaz de reter um grau de controle sobre os dados armazenados localmente
- Redundância
  - os dados podem ser replicados em sites remotos, e o sistema pode funcionar mesmo que um site falhe
- Desvantagem
  - Maior complexidade exigida para garantir coordenação apropriada entre os sites
    - Custo de desenvolvimento de software
    - Maior potencial para bugs
    - Maior sobrecarga do processamento

Questões de implementação para bancos de dados distribuídos

- Atomicidade necessária até mesmo para transações que atualizam dados em vários sites
  - Transação não pode ser confirmada em um site e abortada em outro
- Protocolo de commit em duas fases (2PC) usado para garantir a atomicidade
  - Ideia básica: cada site executa a transação até imediatamente antes do commit, e a decisão final fica para uma coordenador
  - Cada site precisa seguir a decisão do coordenador: mesmo que haja uma falha enquanto espera pela decisão do coordenador
    - Para fazer isso, as atualizações de transação são registradas em log, em armazenamento estável, e a transação é registrada como “aguardando”
- 2PC nem sempre é apropriado: outros modelos de transação baseados em mensagem persistente e fluxos de transação também são usados
- Controle de concorrência distribuído (e detecção de impasse) exigido
- Replicação de itens de dados exigida para melhorar a disponibilidade dos dados

Tipos de redes

- Redes locais (LANs) – compostas de processadores que são distribuídos por pequenas áreas geográficas, como um único prédio ou alguns prédios adjacentes
- Redes remotas (WANs) – compostas de processadores distribuídos por uma grande área geográfica
- Conexão descontínua – WANs, como aquelas baseadas em discagem periódica (usando, por exemplo, UUCP), que estão conectadas apenas por parte do tempo
- Conexão contínua - WANs, como a Internet, onde os hosts estão conectados à rede o tempo todo

Bancos de Dados Paralelos

Bancos de Dados Paralelos

- Máquinas paralelas estão se tornando muito comuns e acessíveis
  - Os preços de microprocessadores, memória e discos têm caído bastante
- Os bancos de dados estão se tornando cada vez maiores
  - grandes volumes de dados de transação são coletados e armazenados para análise posterior.
  - objetos de multimídia (como imagens) estão cada vez mais sendo armazenados em bancos de dados
- Sistemas de banco de dados paralelos em grande escala cada vez mais usados para:
  - armazenar grandes volumes de dados
  - tempo de processamento consumindo consultas de apoio à decisão
  - oferecer throughput alto para processamento de transação

Paralelismo em bancos de dados

- Os dados podem ser particionados entre vários discos para E/S paralela.
- Operações relacionais individuais (por exemplo, sort, join, agregação) podem ser executadas em paralelo
  - Os dados podem ser particionados e cada processador pode trabalhar independentemente em sua própria partição
- As consultas são expressas em linguagem de alto nível (SQL, traduzidas para a álgebra relacional)
  - torna o paralelismo mais fácil
- Diferentes consultas podem rodar em paralelo entre si.
  O controle de concorrência cuida de conflitos
- Assim, os bancos de dados são candidatos naturais ao paralelismo

Paralelismo de E/S

- Reduza o tempo exigido para apanhar relações do disco particionando as relações em vários discos
- Particionamento horizontal
  - As tuplas de uma relação são divididas entre muitos discos, de modo que cada tupla resida em um disco

Técnicas de particionamento (número de discos = n)

- Rodízio:
  - Envie a iª tupla inserida na relação no disco i mod n
- Particionamento de hash:
  - Escolha um ou mais atributos como atributos de particionamento
  - Escolha a função de hash h com intervalo 0...n - 1
  - Considere que i indica o resultado da função de hash h aplicado ao valor do atributo de particionamento de uma tupla. Envie a tupla ao disco i

Paralelismo por intervalo

- Particionamento de intervalo:
  - Escolha um atributo como atributo de particionamento
  - Um vetor de particionamento [$\boldsymbol{v_0}$, $\boldsymbol{v_1}$, ..., $\boldsymbol{v_{n-2}}$] é escolhido
  - Considere v como o valor do atributo de particionamento de uma tupla. Tuplas tais que $\boldsymbol{v_i} \leq \boldsymbol{v_{i+1}}$ vão para o disco I + 1. Tuplas com v < $\boldsymbol{v_0}$ vão para o disco 0 e tuplas com v ≥ $\boldsymbol{v_{n-2}}$ vão para o disco n-1
  - Por exemplo, com um vetor de particionamento [5,11], uma tupla com valor de atributo de particionamento 2 irá para o disco 0, uma tupla com valor 8 irá para o disco 1, enquanto uma tupla com valor 20 irá para o disco 2

Comparação de técnicas de particionamento

- Avalie a facilidade com que as técnicas de particionamento admitem os seguintes tipos de acesso aos dados:
  - 1. Varredura da relação inteira
  - 2. Localização de uma tupla de forma associativa – consultas pontuais
    - Por exemplo, r.A = 25
  - 3. Localização de todas as tuplas tais que o valor de determinado atributo se encontre dentro de um intervalo especificado - consultas de intervalo
    - Por exemplo, $10 \leq r.A < 25$
- Comparar rodízio, hash e intervalo

Análise de particionamento por rodízio

- Vantagens
  - Mais adequada para a varredura sequencial da relação inteira em cada consulta
  - Todos os discos têm quase o mesmo número de tuplas; o trabalho de recuperação é, portanto, bem balanceado entre os discos
- Consultas de intervalo são difíceis de processar
  - Nenhum agrupamento - tuplas espalhadas por todos os discos

Análise de particionamento por hash

- Bom para acesso sequencial
  - Supondo que a função de hash seja boa e os atributos de particionamento formem uma chave, as tuplas serão distribuídas uniformemente entre os discos
  - O trabalho de recuperação é, então, bem balanceado entre os discos
- Bom para consultas pontuais sobre o atributo de particionamento
  - Pode pesquisar único disco, deixando outros disponíveis para responder a outras consultas
  - O índice sobre o atributo de particionamento pode ser local ao disco, tornando a pesquisa e atualização mais eficientes
- Sem agrupamento, de modo que é difícil responder a consultas de intervalo

Análise de particionamento por intervalo

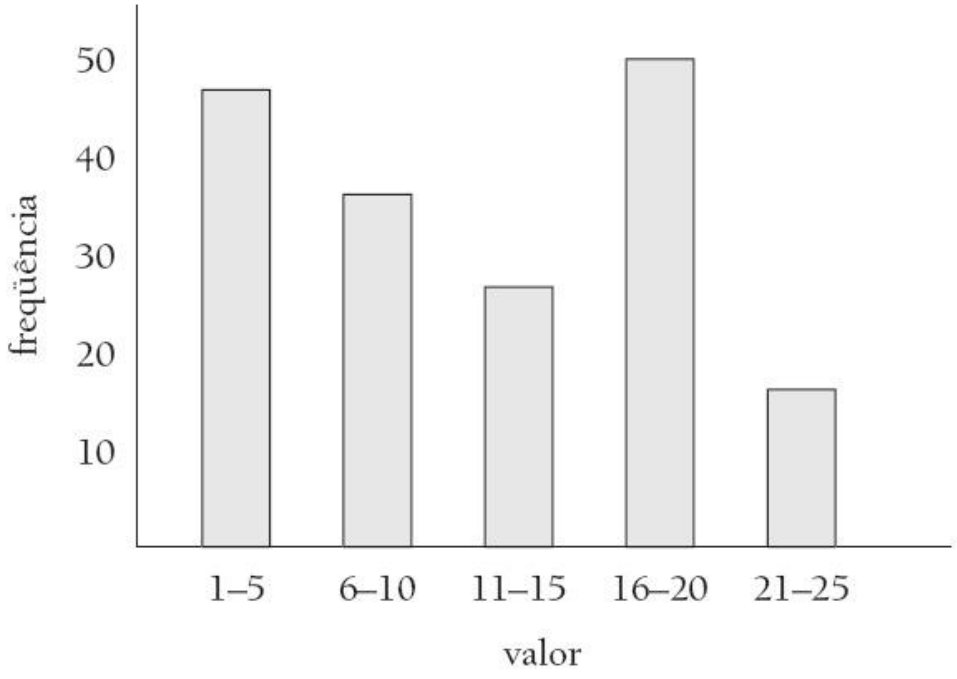
- Oferece agrupamento de dados particionando o valor do atributo
- Bom para acesso sequencial
- Bom para consultas pontuais sobre atributo de particionamento: somente um disco precisa ser acessado
- Para consultas de intervalo sobre o atributo de particionamento, um a alguns discos precisam ser acessados
  - Os discos restantes estão disponíveis para outras consultas
  - Bom se as tuplas de resultado forem de um a alguns blocos
  - Se muitos blocos tiverem que ser apanhados, eles ainda são apanhados de um a alguns discos, e o paralelismo em potencial no acesso ao disco é desperdiçado
    - Exemplo de distorção de execução

Distorção

- A distribuição de tuplas para discos pode ser distorcida — ou seja, alguns discos têm muitas tuplas, enquanto outros podem ter menos tuplas
- Tipos de distorção:
  - Distorção de valor de atributo
    - Alguns valores aparecem nos atributos de particionamento de muitas tuplas; todas as tuplas com o mesmo valor para o atributo de particionamento acabam ficando na mesma partição
    - Pode ocorrer com particionamento de intervalo e particionamento de hash.
  - Distorção de partição
    - Com o particionamento de intervalo, o vetor de partição mal escolhido pode atribuir muitas tuplas a algumas partições e poucas a outras
    - Menos provável com o particionamento de hash se uma boa função de hash for escolhida

Tratamento da distorção usando histogramas

- Vetor de particionamento balanceado pode ser construído a partir do histograma em um padrão relativamente simples
- Considere a distribuição uniforme dentro de cada intervalo do histograma
- O histograma pode ser construído varrendo a relação ou amostrando (blocos contendo) tuplas da relação



Tratamento da distorção usando particionamento de processador virtual

- A distorção no particionamento de intervalo pode ser tratado de forma elegante usando o particionamento de processador virtual:
  - crie uma grande quantidade de partições (digamos, 10 a 20 vezes o número de processadores)
  - Atribua processadores virtuais às partições em padrão de rodízio ou baseado no custo estimado do processamento de cada partição virtual
- Ideia básica:
  - Se qualquer partição normal tiver sido distorcida, é bem provável que a distorção se espalhe por diversas partições virtuais
  - Partições virtuais distorcidas são espalhadas por uma série de processadores, de modo que o trabalho é distribuído uniformemente!

Paralelismo entre consultas

- As consultas/transações são executadas em paralelo entre si
- Aumenta throughput da transação, usado principalmente para ganho de escala de um sistema de processamento de transação par dar suporte a uma grande quantidade de transações por segundo
- Forma mais fácil de paralelismo para dar suporte, principalmente em um banco de dados paralelo de memória compartilhada, pois até mesmo sistemas de banco de dados sequenciais admitem o processamento concorrente
- Mais complicado de implementar em arquiteturas de disco compartilhado ou nada compartilhado
  - O bloqueio e o registro em log precisam ser coordenados pela passagem de mensagens entre os processadores
  - Os dados em um buffer local podem ter sido atualizados em outro processador
  - A coerência de cache precisa ser mantida - leituras e escritas de dados no buffer precisam encontrar a versão mais recente dos dados

Paralelismo intraconsulta

- Execução de uma única consulta em paralelo em múltiplos processadores/discos, importante para agilizar consultas de longa duração
- Duas formas complementares d paralelismo intraconsulta:
  - Paralelismo intraoperação – coloca em paralelo a execução de cada operação individual na consulta
  - Paralelismo entre operações – executa as diferentes operações em uma expressão de consulta em paralelo
- a primeira forma se expande melhor com o aumento do paralelismo, pois o número de tuplas processadas por cada operação normalmente é mais do que o número de operações em uma consulta

Classificação paralela

- Classificação de particionamento de intervalo
- Escolha processadores P_0, ..., P_m, onde m ≤ n-1 para fazer a classificação
- Crie o vetor de partição de intervalo com m entradas, sobre os atributos de classificação
- Redistribua a relação usando particionamento de intervalo
  - todas as tuplas que se encontram no ith intervalo são enviados ao processador P_i
  - P_i armazena as tuplas que recebeu temporariamente no disco D_i
  - Essa etapa exige sobrecarga de E/S e comunicação
- Cada processador P_i classifica sua partição da relação localmente
- Cada processador executa a mesma operação (classificação) em paralelo com outros processadores, sem qualquer interação com os outros (paralelismo de dados)
- A operação de mesclagem final é trivial: o particionamento de intervalo garante que os valores de chave no processador P_i são todos menores que os valores de chave em P_j (i < j)

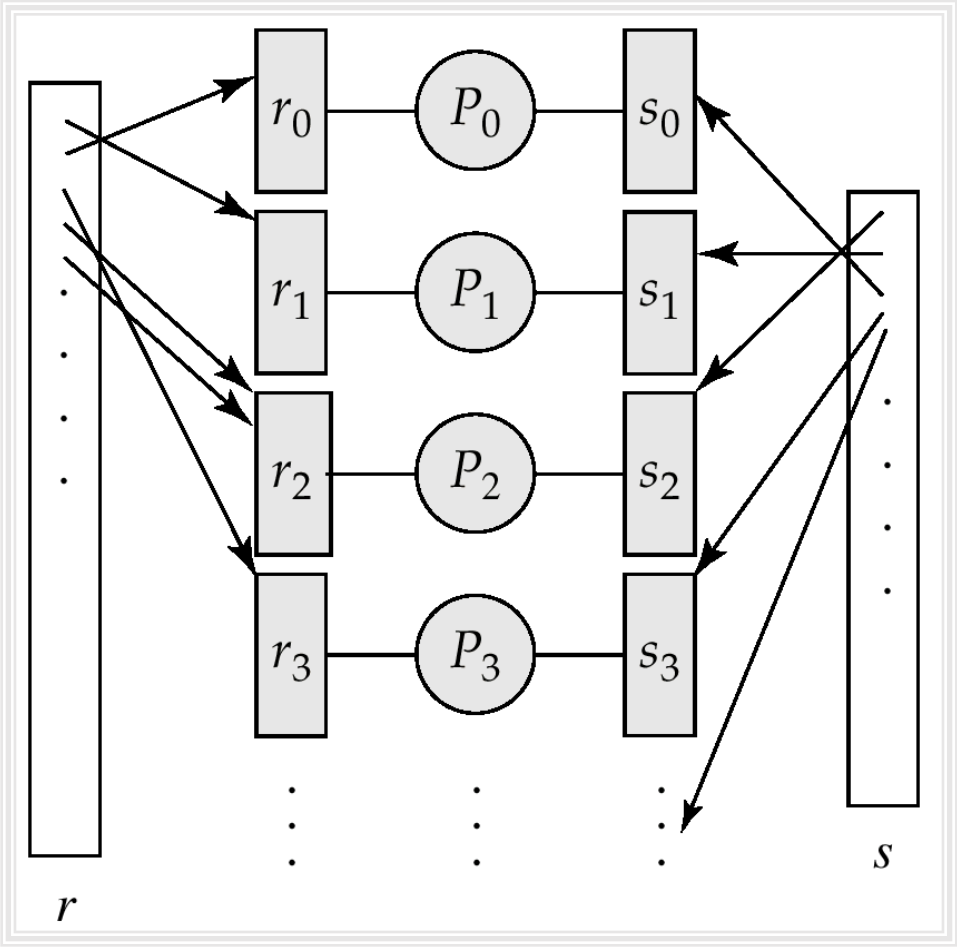
Junção paralela

- A operação de junção exige pares de tuplas a serem testadas para ver se satisfazem a condição de junção e, nesse caso, o par é acrescentado à saída da junção
- Algoritmos de junção paralela tentam dividir os pares a serem testados por vários processadores. Cada processador, então, calcula parte da junção localmente
- Em uma etapa final, os resultados de cada processador podem ser coletados juntos para produzir o resultado final

Junção particionada

- Para equijunções e junções naturais, é possível particionar as duas relações de entrada entre os processadores, e calcular a junção localmente em cada processador.
- Considere que R e S sejam as relações de entrada, e queremos calcular $\boldsymbol{R_{R.a=S.b}} \bowtie \boldsymbol{S}$
- R e S são particionados em n partições, indicadas como $\boldsymbol{R_0}$, $\boldsymbol{R_1}$, ..., $\boldsymbol{R_{n-1}}$ e $\boldsymbol{S_0}$, $\boldsymbol{S_1}$, ..., $\boldsymbol{S_{n-1}}$.
- Pode usar particionamento de intervalo ou particionamento de hash.
  - R e S precisam ser particionados em seus atributos de junção $\boldsymbol{R.a}$ e $\boldsymbol{S.b}$, usando o mesmo vetor de particionamento de intervalo ou função de hash
- As partições $\boldsymbol{R_i}$ e $\boldsymbol{S_i}$ são enviadas ao processador $\boldsymbol{P_i}$,
- Cada processador $\boldsymbol{P_i}$ calcula localmente $\boldsymbol{R_i}_{\boldsymbol{R.a=S.b}} \bowtie \boldsymbol{S_i}$. Quaisquer métodos de junção padrão podem ser usados.

Junção particionada (cont.)



Paralelismo entre operadores

- Paralelismo canalizado
  - Considere uma junção de quatro relações
    - $\boldsymbol{R_1 \bowtie R_2 \bowtie R_3 \bowtie R_4}$
  - Monte uma pipeline que calcule as três junções em paralelo:
    - Considere que $\boldsymbol{P_1}$ tenha sido atribuído ao cálculo de $\boldsymbol{temp_1 = R_1 \bowtie R_2}$
    - E $\boldsymbol{P_2}$ seja atribuído ao cálculo de $\boldsymbol{temp_2 = R_2 \bowtie R_3}$
    - E $\boldsymbol{P_3}$ seja atribuído ao cálculo de $\boldsymbol{temp_3 = R_3 \bowtie R_4}$
  - Cada uma dessas operações pode ser executada em paralelo, enviando tuplas de resultado que ela calcula para a operação seguinte, mesmo quando está calculando outros resultados
    - Desde que seja usado um algoritmo de avaliação de junção passível de pipeline (por exemplo: junção de loops aninhados indexados)

Fatores limitando a utilidade do paralelismo canalizado

- O paralelismo em pipeline é útil porque evita a escrita de resultados intermediários em disco
- Útil com pequena quantidade de processadores, mas não se expande bem com mais processadores. Um motivo é que as cadeias de pipeline não conseguem tamanho suficiente
- Não pode canalizar operadores que não produzem saída até que todas as entradas tenham sido acessadas (por exemplo: agregação e classificação)
- Pouco ganho de velocidade é obtido para casos frequentes de distorção em que o custo de execução de um operador é muito maior que os outros.

Banco de Dados Distribuídos

Sistema de banco de dados distribuído

- Um sistema de banco de dados distribuído consiste em sites pouco acoplados, que não compartilham um componente físico
- Os sistemas de banco de dados que executam em cada site são independentes um do outro
- As transações podem acessar dados em um ou mais sites

Bancos de dados distribuídos homogêneos

- Em um banco de dados distribuído homogêneo
  - Todos os sites possuem software idêntico
  - Estão cientes um do outro e concordam em cooperar no processamento de solicitações do usuário.
  - Cada site entrega parte de sua autonomia em termos de direito de mudar esquemas ou software
  - Aparece ao usuário como um único sistema
- Em um banco de dados distribuído heterogêneo
  - Diferentes sites podem usar diferentes esquemas e software
    - A diferença no esquema é um grande problema para o processamento da consulta
    - A diferença no software é um grande problema para o processamento da transação
  - Os sites podem não estar cientes um do outro e só podem oferecer facilidades limitadas para cooperação no processamento da transação

Armazenamento de dados distribuído

- Considere o modelo de dados relacional
- Replicação
  - O sistema mantém várias cópias dos dados, armazenadas em diferentes sites, para a recuperação mais rápida e tolerância a falhas
- Fragmentação
  - A relação é particionada em vários fragmentos armazenados em sites distintos
- Replicação e fragmentação podem ser combinadas
  - A relação é particionada em vários fragmentos: o sistema mantém várias réplicas idênticas de cada um desses fragmentos

Replicação de dados

- Uma relação ou fragmento de uma relação é replicado se for armazenado redundantemente em dois ou mais sites
- A replicação total de uma relação é o caso onde a relação é armazenada em todos os sites
- Bancos de dados totalmente redundantes são aqueles em que cada site contém uma cópia do banco de dados inteiro

Replicação de dados (cont.)

- Vantagens da replicação
  - Disponibilidade: a falha do site contendo a relação $\boldsymbol{R}$ não resulta na indisponibilidade de $\boldsymbol{R}$ se houver réplicas
  - Paralelismo: as consultas sobre $\boldsymbol{R}$ podem ser processadas por vários nós em paralelo
  - Transferência de dados reduzida: a relação $\boldsymbol{R}$ está disponível localmente em cada site contendo uma réplica de $\boldsymbol{R}$
- Desvantagens da replicação
  - Custo aumentado das atualizações: cada réplica da relação $\boldsymbol{R}$ precisa ser atualizada
  - A complexidade cada vez maior do controle de concorrência: atualizações concorrentes para réplicas distintas podem levar a dados inconsistentes, a menos que mecanismos especiais de controle de concorrência sejam implementados
    - Uma solução: escolha uma cópia como cópia primária e aplique operações de controle de concorrência na cópia primária

Fragmentação de dados

- Divisão da relação $\boldsymbol{R}$ em fragmentos $\boldsymbol{R_1}$, $\boldsymbol{R_2}$, ..., $\boldsymbol{R_n}$, que contêm informações suficientes para reconstruir a relação $\boldsymbol{R}$
- Fragmentação horizontal: cada tupla de $\boldsymbol{R}$ é atribuída a um ou mais segmentos
- Fragmentação vertical: o esquema para a relação $\boldsymbol{R}$ é dividido em vários esquemas menores
  - Todos os esquemas precisam conter uma chave candidata comum (ou superchave) para garantir a propriedade de junção sem perdas.
  - Um atributo especial, o atributo id-tupla, pode ser acrescentado a cada esquema para servir como chave candidata.

Fragmentação horizontal da relação conta
Esquema-conta = (nomeAgencia, numeroConta, saldo)

nomeAgencia	numeroConta	Saldo
Hillside	A-305	500
Hillside	A-226	336
Hillside	A-155	62

$$conta_1 = \sigma_{nomeAgencia="Hillside"}(conta)$$

nomeAgencia	numeroConta	Saldo
Valleyview	A-177	205
Valleyview	A-402	10000
Valleyview	A-408	1123
Valleyview	A-639	750

$$conta_2 = \sigma_{nomeAgencia="Valleyview"}(conta)$$

Fragmentação vertical da relação InfoFuncionario (nomeAgencia, nomeCliente, numeroConta, saldo, id)

nomeAgencia	nomeCliente	id
Hillside	Lowman	1
Hillside	Camp	2
Valleyview	Camp	3
Valleyview	Kahn	4
Hillside	Kahn	5

$$infoFuncionario_1 = \pi_{nomeAgencia,nomeCliente,id}(infoFuncionario)$$

numeroConta	saldo	id
A-177	205	1
A-402	10000	2
A-408	1123	3
A-639	750	4
A-747	5000	5

$$infoFuncionario_2 = \pi_{numeroConta,saldo,id}(infoFuncionario)$$

Vantagens da fragmentação

- Horizontal:
  - permite o processamento paralelo sobre fragmentos de uma relação
  - permite que uma relação seja dividida de modo que as tuplas sejam localizadas onde são acessadas com mais frequência
- Vertical:
  - permite que as tuplas sejam divididas de modo que cada parte da tupla seja armazenada onde é acessada com mais frequência
  - o atributo id-tupla permite a junção eficiente de fragmentos verticais
  - permite o processamento paralelo sobre uma relação
- A fragmentação vertical e horizontal podem ser misturadas
  - Fragmentos podem ser sucessivamente fragmentados a uma profundidade qualquer

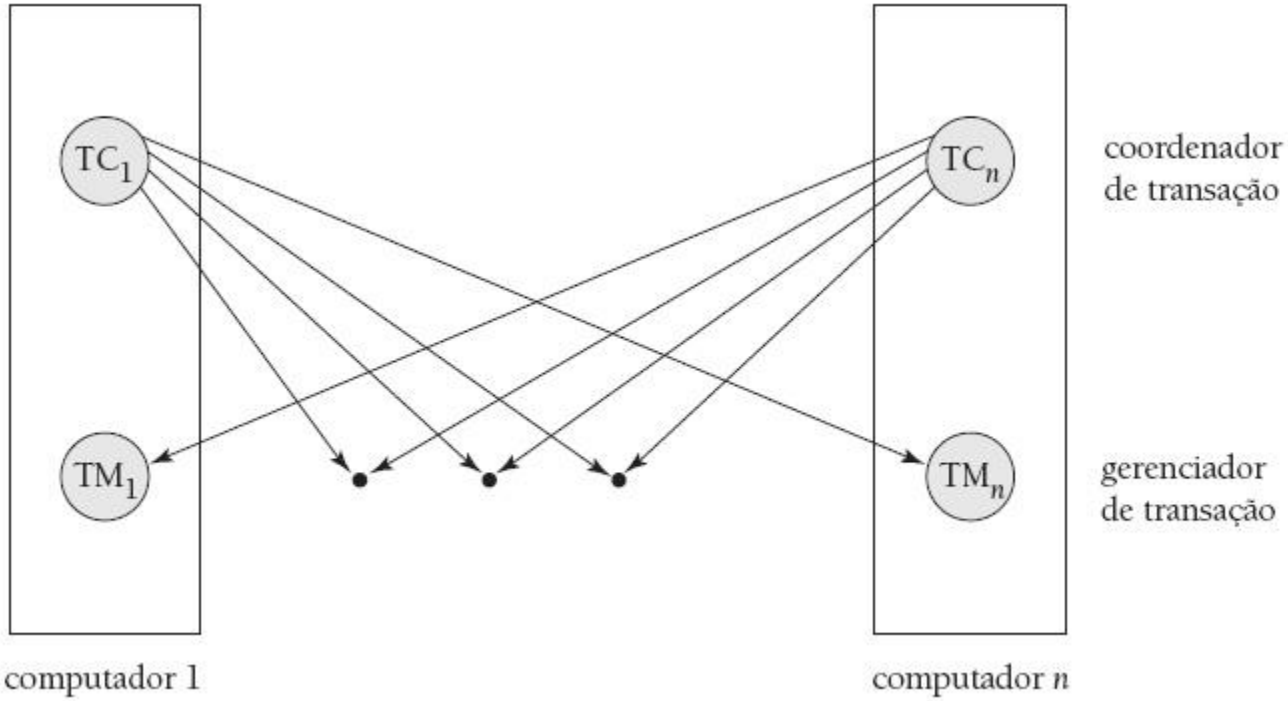
Transparência de dados

- Transparência de dados:
- Transparência de dados: Grau ao qual o usuário do sistema pode permanecer desavisado dos detalhes de como e onde os itens de dados são armazenados em um sistema distribuído
- Considere questões de transparência em relação a:
  - Transparência de fragmentação
  - Transparência de replicação
  - Transparência de local

Transações distribuídas

- A transação pode acessar dados em vários sites
- Cada site possui um gerenciador de transação, responsável por:
  - Manter um log para fins de recuperação
  - Participar na coordenação da execução concorrente das transações em execução nesse site
- Cada site possui um coordenador de transação, responsável por:
  - Iniciar a execução das transações que originam no site
  - Distribuir sub-transações nos sites apropriados para execução
  - Coordenar o término de cada transação que origina no site, que pode resultar na transação sendo confirmada em todos os sites ou abortada em todos os sites

Arquitetura de sistema de transação



Falhas exclusivas aos sistemas distribuídos

- Falha de um site
- Perda de mensagens
  - Tratada por protocolos de controle de transmissão de rede, como TCP-IP
- Falha de um enlace de comunicação
  - Tratada por protocolos de rede, roteando mensagens por meio de enlaces alternativos
- Partição de rede
  - Uma rede é considerada particionada quando tiver sido dividida em dois ou mais subsistemas que faltam qualquer conexão entre eles
    - Nota: um subsistema pode consistir em um único nó
- Particionamento de rede e falhas do site geralmente não são distinguidos

Protocolos de commit

- Protocolos de commit são usados para garantir a atomicidade entre os sites
  - uma transação que é executada em vários sites precisa ser confirmada em todos os sites ou abortada em todos os sites
  - não é aceitável ter uma transação confirmada em um site e abortada em outro
- O protocolo de commit em duas fases (2PC) é muito utilizado
- O protocolo de commit em três fases (3PC) é mais complicado e mais dispendioso, mas evita alguma desvantagens do protocolo de commit em duas fases

Protocolo de commit em duas fases (2PC)

- Considere o modelo falhar-parar - os sites que falham simplesmente deixam de trabalhar e não causam qualquer outro prejuízo, como enviar mensagens incorretas a outros sites.
- A execução do protocolo é iniciada pelo coordenador após a última etapa da transação ter sido alcançada.
- O protocolo envolve todos os sites locais em que a transação foi executada
- Considere que $\boldsymbol{T}$ seja uma transação iniciada no site $\boldsymbol{S_i}$ e considere que o coordenador de transação em $\boldsymbol{S_i}$ seja $\boldsymbol{C_i}$

Fase 1: Obtendo uma decisão

- O coordenador pede a todos os participantes para que se preparem para confirmar a transação $\mathbf{T_i}$.
  - $\mathbf{C_i}$ acrescenta os registros <prepare $\boldsymbol{T}$> ao log e força o log para o armazenamento estável
  - envia mensagens prepare $\boldsymbol{T}$ a todos os sites em que $\boldsymbol{T}$ foi executado
- Ao receber a mensagem, o gerenciador de transação no site determina se pode confirmar a transação
  - se não, acrescente um registro <no $\boldsymbol{T}$> ao log e envie mensagem abort $\boldsymbol{T}$ para $\mathbf{C_i}$
  - se a transação puder ser confirmada, então:
  - acrescente o registro <ready $\boldsymbol{T}$> ao log
  - force todos os registros para $\boldsymbol{T}$ ao armazenamento estável
  - envie mensagem <ready $\boldsymbol{T}$> para $\mathbf{C_i}$

Fase 2: Registrando a decisão

- T pode ser confirmado se $\mathbf{C_i}$ recebeu uma mensagem <ready $\boldsymbol{T}$> de todos os sites participantes: caso contrário, $\boldsymbol{T}$ precisa ser abortado
- O coordenador acrescenta um registro de decisão, <commit $\boldsymbol{T}$> ou <abort $\boldsymbol{T}$>, ao log e força o registro para o armazenamento estável. Uma vez no armazenamento estável, o registro é irrevogável (mesmo se houver falhas)
- O coordenador envia uma mensagem a cada participante informando sobre a decisão (commit ou abort)
- Os participantes tomam a ação apropriada no local

Tratamento de falhas - falha no site

- Quando o site $\mathbf{S_i}$ se recupera, ele examina seu log para determinar o destino das transações ativas no momento da falha.
- Log contém registro <commit $\boldsymbol{T}$>: site executa redo ($\boldsymbol{T}$)
- Log contém registro <abort $\boldsymbol{T}$>: site executa undo ($\boldsymbol{T}$)
- Log contém registro <ready $\boldsymbol{T}$>: site deve consultar $\mathbf{C_i}$ para determinar o destino de T.
  - Se T confirmado, redo ($\boldsymbol{T}$)
  - Se T abortado, undo ($\boldsymbol{T}$)
- O log não contém registros de controle referentes a respostas de $\boldsymbol{T}$ de que $\mathbf{S_k}$ falhou antes de responder à mensagem prepare $\boldsymbol{T}$ de $\mathbf{C_i}$
  - como a falha de $\mathbf{S_k}$ impede o envio de tal resposta, $\mathbf{C_i}$ deverá abortar $\boldsymbol{T}$
  - $\mathbf{S_k}$ deverá executar undo ($\boldsymbol{T}$)

Tratamento de falhas – falha do coordenador

- Se o coordenador falhar enquanto o protocolo de commit para T está executando, então os sites participantes terão que decidir sobre o destino de T :
  - Se um site ativo tiver um registro <commit T > em seu log, então T terá que ser confirmado
  - Se um site ativo tiver um registro <abort T > em seu log, então T terá que ser abortado
  - Se algum site participante ativo não tiver um registro <ready T > em seu log, então o coordenador que falhou C_i não pode ter decidido confirmar T. Portanto, pode abortar T
  - Se nenhum dos casos acima for mantido, então todos os sites ativos deverão ter um registro <ready T> em seus logs, mas nenhum registro de controle adicional (como <abort T> de <commit T>). Nesse caso, os sites ativos terão que esperar que C_i se recupere para encontrar a decisão
- Problema de bloqueio: os sites ativos podem ter que esperar a recuperação do coordenador que falhou

Tratamento de falhas - partição de rede

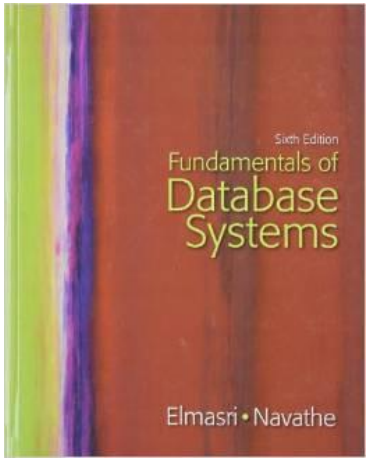
- Se o coordenador e todos os seus participantes permanecerem em uma partição, a falha não terá efeito sobre o protocolo de commit
- Se o coordenador e seus participantes pertencerem a várias partições:
  - Os sites que não estão na partição contendo o coordenador acham que o coordenador falhou e executam o protocolo para tratar da falha do coordenador
    - Nenhum dano acontece, mas os sites terão que esperar pela decisão do coordenador
- O coordenador e os sites estão na mesma partição quando o coordenador pensa que os sites na outra partição falharam, e segue o protocolo normal de commit
    - Novamente, nenhum dano acontece

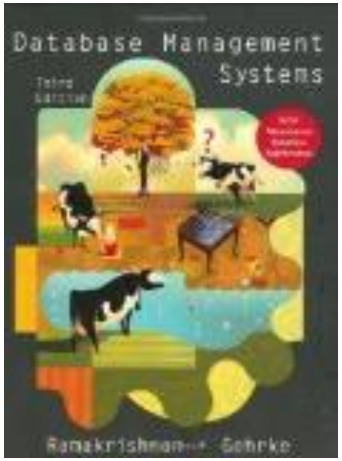
- Transações em dúvida têm um registro de log <ready T>, mas sem <commit T > nem <abort T >
- O site em recuperação precisa determinar o status commit-abort de tais transações contatando outros sites; isso pode atrasar e potencialmente bloquear a recuperação
- Os algoritmos de recuperação podem anotar informações de bloqueio no log
  - Em vez de <ready T>, escreva <ready T, L>
    - L = lista de bloqueios mantidos por T quando o log é escrito (bloqueios de leitura podem ser omitidos)
  - Para cada transação em dúvida T, todos os bloqueios anotados no registro de log <ready T, L> são readquiridos
- Após a reaquisição do bloqueio, o processamento da transação pode retomar; o commit ou rollback de transações em dúvida é realizado simultaneamente com a execução de novas transações

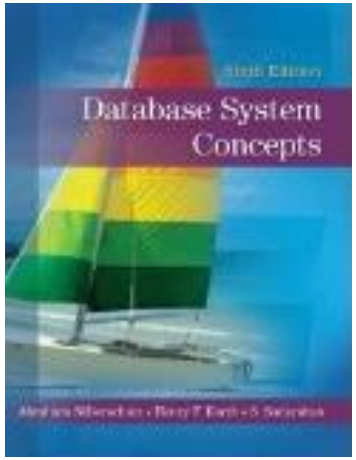
Controle de concorrência

- Adaptações dos esquemas de controle de concorrência para uso no ambiente distribuído
- Considera-se que cada site participa na execução de um protocolo de commit para garantir a atomicidade global da transação
- Consideramos que todas as réplicas de qualquer item estão atualizadas

Referências







A maioria dos slides apresenta exemplos do livro Database System Concepts