2
Conceitos

Neste capítulo apresentam-se alguns conceitos relacionados a bancos de dados paralelos, processamento de consulta adaptativo e grades computacionais, que são necessários ao entendimento deste trabalho.

2.1. Banco de Dados Paralelos

Os Sistemas de Gerência de Banco de Dados Paralelos (SGBDP) surgiram no final da década de 80 como uma alternativa para solucionar o problema do "gargalo de E/S" dos bancos de dados convencionais. Tal problema é decorrente do alto custo de acesso aos mecanismos de armazenamento secundário (i.e. discos) em comparação à memória principal. Uma solução para este problema é o particionamento dos dados em um conjunto de discos que possam ser acessados paralelamente durante a execução de uma consulta. Tal proposta possibilitaria a elevação do *throughput* do sistema para a soma do *throughput* de cada disco utilizado.

As definições de banco de dados paralelos e banco de dados distribuídos algumas vezes se confundem na literatura, uma vez que ambas tratam da questão do particionamento, mas algumas diferenças devem ser observadas:

- Os SGBDPs são fortemente acoplados (existe um único ponto de controle que tem a função de dividir a consulta em fragmentos e as executar de forma paralela) enquanto os SGBDDs são fracamente acoplados.
- Os SGBDDs distribuem os dados segundo sua utilização (princípio da localidade), já nos SGBDPs a distribuição é dependente da arquitetura dos sistema.
    - Os SGBDDs são adequados para aumentar a robustez e a disponibilidade dos dados e os SGBDPs são mais adequados a escalabilidade do sistema.

- Os SGBDDs endereçam questões de autonomia, distribuição e heterogeneidade entre os componentes.
- Os SGBDD por sua origem distribuída podem levar ao paralelismo.

2.1.1. Métricas para Sistemas Paralelos

Em (D.J.DEWITT & J.GRAY, 1992) apresenta-se a aceleração (*speedup*) e escalabilidade (*scaleup*) como métricas para o desempenho de um sistema paralelo. A aceleração descreve a habilidade que um sistema possui para resolver um problema quando se varia sua capacidade de processamento, ou seja, o ganho que se obtém ao executar um mesmo problema em sistemas com diferentes capacidades de processamento e pode ser medida como:

$$Aceleração = \frac{TempoDecorridoSistemaMenor}{TempoDecorridoSistemaMaior}$$

Em uma situação ideal, a aceleração deveria ser linear em função da capacidade de processamento que se adicionou ao sistema, no entanto ela nem sempre é possível. Nestes casos, uma medida conhecida como eficiência, poderia informar quanto esforço o sistema desperdiça em sincronização e comunicação. Ela pode ser calculada como:

$$Eficiência = \frac{Aceleração}{CapacidadeSistemaMaior}$$

Já a escalabilidade descreve o comportamento do sistema em função do crescimento de ambos, da capacidade de processamento e da complexidade do problema e indica o potencial de crescimento da solução.

$$\text{Escalabilidade} = \frac{\text{TempoDecorridoProblemaMenorSistemaMenor}}{\text{TempoDecorridoProblemaMaiorSistemaMaior}}$$

Na prática alguns fatores impedem que as soluções possuam aceleração linear, tais problemas são conhecidos como:

- Tempo de Inicialização: O tempo gasto para iniciar um processamento paralelo. Se milhares de processos devem ser iniciados, isso poderia dominar o custo de execução.
- Interferência: A redução de velocidade que se tem quando vários processos devem acessar um recurso compartilhado.
- Desequilíbrio: ao se usar vários processos para realizar uma operação, é natural que o tamanho médio da operação realizada por cada um diminua, no entanto, se não houver um balanceamento de carga adequado à variância pode ser maior que a média e dominar o custo de execução.

Na seção a seguir apresentam-se algumas arquiteturas de hardware para sistema paralelo e discutem-se os problemas associados a cada uma em função dos problemas de interferência e balanceamento de carga.

2.1.2. Arquiteturas de Hardware para Sistemas Paralelos

Tradicionalmente, as arquiteturas de hardware para banco de dados paralelos podem ser classificadas em: memória compartilhada, disco compartilhado e nada compartilhado.

Em uma arquitetura de memória compartilhada todos os processadores têm acesso a qualquer módulo de memória ou disco através de uma interconexão rápida. Esta arquitetura apresenta como vantagens a simplicidade de implementação e o balanceamento de carga que pode ser obtido dinamicamente. Por outro lado, esta arquitetura apresenta problemas como: elevado custo e baixa escalabilidade. O custo é decorrente da necessidade de um mecanismo de interconexão complexo que deve vincular cada processador a cada módulo de memória ou disco. E a baixa escalabilidade ocorre em função da interferência criada na rede de interconexão que, consequentemente, diminui o desempenho do sistema.

Na arquitetura de disco compartilhado cada processador possui sua própria memória principal e as unidades de disco são compartilhadas. Seu custo é bem menor se comparado à arquitetura de memória compartilhada, pois a tecnologia de comunicação utilizada na interconexão pode ser um simples barramento. Esta arquitetura é ideal para sistemas de banco de dados em que atualizações não sejam

realizadas com freqüência, uma vez que um mecanismo de bloqueio mais complexo deve ser utilizado para garantir acesso único à pagina a ser atualizada.

Por fim, na arquitetura nada compartilhado cada processador possui sua própria memória principal e unidades de disco. Assim, cada nó pode ser visto como um site de um sistema de banco de dados distribuídos e, desta forma, a maioria das soluções projetadas para este ambiente (i.e. fragmentação, gerenciamento de transação e processamento distribuído) pode ser reutilizada. As principais virtudes desta arquitetura são: o custo e a escalabilidade. A vantagem do custo é a mesma que a da arquitetura de disco compartilhado e, a escalabilidade pode ser obtida adotando um critério de particionamento cuidadoso dos dados. Em contrapartida, o principal problema desta arquitetura é o balanceamento de carga que depende do particionamento adotado para as cargas de trabalho das consultas. Além disso, a adição de nós poderia provocar uma reorganização na distribuição dos dados.

Segundo (D.J.DEWITT & J.GRAY, 1992) a arquitetura do tipo nada compartilhado seria a melhor para sistemas de banco de dados paralelos. O principal argumento na defesa desta arquitetura é a redução da interferência criada com o aumento do número de processadores e, desta forma, uma maior escalabilidade poderia ser obtida. Outro argumento é que nas demais arquiteturas um volume de dados muito grande é movimentado na rede de interconexão, enquanto que nesta são enviadas apenas “perguntas e respostas”.

Em (M.G.NORMAN, T.ZUREK et al., 1996) são feitas algumas críticas quanto a afirmação de que haveria um consenso na utilização da arquitetura nada compartilhado realizada em (D.J.DEWITT & J.GRAY, 1992). Para os autores, o ideal seria uma arquitetura híbrida em que os benefícios de ambas as arquiteturas fossem exploradas. Seguindo esta linha, dois novos tipos de arquiteturas foram propostas: a hierárquica e NUMA.

2.1.3. Técnicas de SGBDPs

Nesta seção apresentam-se algumas técnicas que podem ser utilizadas no desenvolvimento de sistemas de banco de dados paralelos. Tais técnicas incluem desde o particionamento dos dados que, pode resolver o problema do “gargalo de E/S” até o processamento de consulta neste ambiente.

2.1.3.1. Particionamento

No contexto da colocação dos dados em banco de dados paralelos três estratégias se destacam: O particionamento por rodízio, *hash* e por intervalo.

O particionamento de rodízio (*round robin*) é a estratégia mais simples e assegura uma distribuição uniforme dos dados. Supondo n partições, a i-ésima tupla a ser inserida seria alocada a partição (i mod n). Este tipo de particionamento é ideal para aplicações que necessitem acessar todos os dados da tabela de forma seqüencial em suas consultas, porém para predicados associativos (i.e. uma consulta procurando por Smith em uma agenda telefônica) esta estratégia não é eficiente.

O particionamento em *hash* realiza a alocação dos dados a partir da aplicação de uma função de *hash* sobre um atributo da relação. Esta estratégia é ideal para aplicações que realizam acesso seqüencial e/ou associativo. Neste último, quando existe um predicado de igualdade sobre o atributo de particionamento da relação apenas um nó precisará ser acessado.

O particionamento de intervalo distribui as tuplas com base nos intervalos de valores de algum atributo da relação. Além de permitir consultas associativas como no *hash*, ela é bem adaptada a consultas de intervalos. Uma característica importante desta estratégia é que ela permite agrupar dados relacionados em uma mesma partição e, desta forma favorece algumas otimizações no acesso aos dados.

2.1.3.2. Tipos de Paralelismo

No contexto de execução de consultas paralelas dois tipos de paralelismo podem ser observados: o interconsulta e o intraconsulta. O paralelismo interconsulta permite a execução de consultas concorrentes originadas de diferentes transações concorrentemente e, tem o objetivo de maximizar o *throughput* das transações. Já o paralelismo intraconsulta permite minimizar o tempo de execução de uma consulta e, pode ser classificado em inter-operador ou intra-operador.

O paralelismo intra-operador é baseado na utilização de várias instâncias do operador, cada instância processa um conjunto de dados distinto e, o particionamento dos dados pode ocorrer de forma estática ou dinâmica.

Normalmente este tipo de paralelismo se beneficia do particionamento inicial dos dados. As operações unárias podem ser facilmente decompostas, para isso basta adicionar uma instância do operador em cada partição. Já as operações binárias (i.e. junções) são mais difíceis de serem implementadas, pois podem necessitar que os dados sejam distribuídos.

O paralelismo inter-operador, por sua vez, pode ser classificado em: paralelismo de canal (*pipeline*) e independente. No paralelismo em canal, vários operadores com o vínculo produtor-consumidor podem ser executados em paralelo e, sua vantagem é permitir que os resultados intermediários não sejam persistidos. O paralelismo independente ocorre quando não há nenhuma dependência entre os operadores relacionados em paralelo e, é bastante atraente, pois não existe a presença de interferência entre os operadores.

2.1.3.3. Processamento de Consulta

Os algoritmos paralelos para a implementação das operações da álgebra relacional formam a base para a construção de máquinas de execução de consultas paralela. Neste sentido, as operações unárias podem ser facilmente implementadas, pois pode ser alcançada pela adição do operador nas partições utilizadas. As operações binárias são mais complexas de serem implementadas, no entanto todas as operações deste tipo podem se basear nas técnicas utilizadas para os algoritmos de junção, pois, podem ser tratadas de maneira bastante semelhante (KJELL BRATBERGSENGEN, 1984).

A escolha da melhor técnica de processamento de consultas em paralelo está intimamente relacionada ao particionamento de dados, pois deve ser alcançado um compromisso entre paralelismo e o custo de comunicação envolvido.

Em (M.T.ÖZSU & P.VALDURIEZ, 1999) apresenta-se três algoritmos para o processamento de junções: algoritmo de *loop* aninhado paralelo (ALAP), de junção associativa paralela (AJAP) e de junção paralela (AJP).

O algoritmo ALAP é baseado na distribuição de todos os fragmentos da segunda relação para os nós onde se encontram os fragmentos da primeira relação. Este algoritmo é ideal para utilização em redes que oferecem suporte ao protocolo *multicast*, pois seu custo de comunicação é elevado. Sua vantagem é poder ser

utilizado para todos os tipos de predicado de junção e não depender do particionamento realizado inicialmente.

O AJAP pode ser utilizado para a realização de junções com predicado de equivalência na qual uma das relações encontra-se dividida pelo atributo de junção. Seja R >< S uma junção na qual R encontra-se particionada de acordo com o predicado de junção. Neste caso, o algoritmo enviará a relação S de forma associativa para os nós de R, com base em uma função de hash aplicada sobre o valor do atributo de junção. Isso garante que uma tupla de S com valor de hash "v" será enviada apenas aos nós de R que também possuam tuplas com valor de hash "v".

O algoritmo AJP pode ser visto como uma generalização do AJAP no qual as duas relações são distribuídas para um conjunto de nós através da aplicação de uma mesma função de hash. Sua vantagem é ser independente do particionamento inicial das relações.

2.1.3.4. Otimização de Consultas

A otimização de consulta está relacionada ao processo de seleção do melhor plano de execução, que minimize a função de custo de objetivo. Um aspecto importante deste ambiente é que o custo da execução de consultas paralelas é maior, mas normalmente seu tempo de execução é reduzido. Tradicionalmente o processo de otimização possui três componentes: o espaço de busca, um modelo de custo e uma estratégia de pesquisa. O espaço de busca descreve o conjunto de planos alternativos que representam a consulta. Estes planos são equivalentes no sentido de que produzem o mesmo resultado, mas difere na ordem com que os operadores são executados. O modelo de custo associa um custo a cada plano e, a estratégia de busca define uma maneira para explorar todos os planos. Geralmente, a otimização de consultas paralelas é mais complexa, pois também deve realizar a alocação de processadores.

Na otimização de consultas paralelas o espaço de busca é bem maior se comparado ao de uma execução de consulta centralizada, na qual apenas um operador é executado por vez, e o formato do plano está restrito a árvores lineares. A definição do espaço de busca deve considerar qual o tipo de paralelismo se deseja empregar, pois dependendo da técnica a ser utilizada alguns formatos de

planos se adequam melhor. Por exemplo, para a execução de consultas com paralelismo independente as árvores fechadas são mais eficientes, enquanto que para a execução em pipeline as árvores lineares apresentam melhor desempenho. A figura 2 ilustra o a forma dessas árvores, na árvore linear as junções são realizadas aos pares, enquanto que na árvore fechada mais de uma operação de junção pode ser realizada ao mesmo tempo.

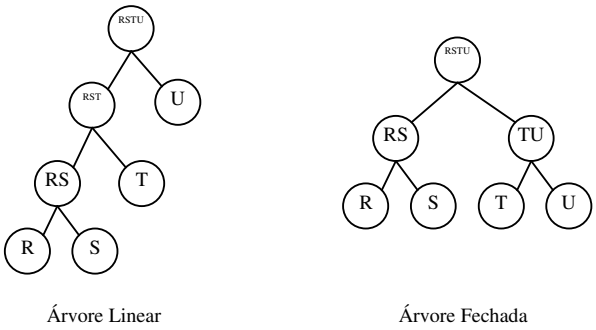

Figura 2. Formato de árvore de otimização

A escolha do melhor algoritmo de pesquisa a ser utilizado depende de quais os formatos de árvores são considerados. Quando são utilizadas apenas árvores lineares o espaço de busca é menor e podem ser empregados algoritmos baseados em programação dinâmica. Esta solução pode utilizar um conjunto de heurísticas para varrer o espaço de busca e garantem que o ótimo vai ser escolhido. Quando as árvores fechadas também são consideradas o espaço de busca aumenta de forma significativa e, neste contexto, são utilizados algoritmos randômicos (ROSANA S.G.LANZELOTTE, P.VALDURIEZ et al., 1993) (Y.IOANNIDIS & E.WONG, 1987) na busca pelo melhor plano. Estes algoritmos garantem que na média um bom plano vai ser escolhido. Em ambos os casos são utilizados um modelo de custo para estimar o custo dos planos. No contexto de execução paralelas os modelos de custo também devem considerar métricas para o custo de comunicação.

Com relação a alocação dos processadores podem ser utilizadas duas estratégias: em uma ou duas fases. Na estratégia de otimização em uma fase todas as possíveis alocações são representadas pelo espaço de busca e todos os formatos de árvores são considerados, já na estratégia de duas fases um plano é criado a partir do processo de otimização tradicional e a partir daí é realizada uma

alocação. Considerando o compromisso que deve haver entre o tempo de otimização e execução e a complexidade envolvida na estratégia em uma fase, a maioria dos trabalhos encontrados utiliza a otimização em duas fases.

2.1.4. Problemas Relacionados ao Ambiente Paralelo

Vários problemas podem ocorrer no processamento de consultas paralelas e aumentar o tempo de execução. Nesta seção apresenta-se alguns dos problemas mais comuns neste ambiente.

O custo de inicialização do ambiente para a execução de consulta paralelas é relativamente alto se comparado a uma execução centralizada. Esta fase inclui a criação dos processos, seu encadeamento e em alguns casos a distribuição de dados. O custo desta fase é proporcional ao grau de paralelismo utilizado e pode dominar o custo de execução de consultas de baixa complexidade.

Outro problema relativo à execução de consultas paralelas é a existência de interferência, que ocorre quando vários processadores tentam acessar um recurso de *hardware* ou *software*. Este problema é bastante comum nas arquiteturas de disco ou memória compartilhada, pois, o número de processadores que concorrem pelo barramento é maior. Uma solução para este problema é duplicar os recursos compartilhados. Quanto à interferência de *software*, um problema clássico é o acesso às estruturas internas do banco de dados como, por exemplo, os índices. Neste âmbito, uma solução possível é a divisão das estruturas e dados em pequenas partições independentes, que são controladas por variáveis de exclusão mútua separadas.

O *skew* é um problema relacionado ao particionamento estático ou dinâmico dos dados. O tempo total de execução de um conjunto de operadores é definido pelo operador com maior tempo, sendo assim é extremamente importante que haja um equilíbrio na distribuição dos dados. Em (C.B.WALTON, R.M.JENEVIN et al., 1991) é apresentado um conjunto de distorções que levam a este problema, entre eles destacam-se: distorção em função dos valores dos atributos, distribuição dos dados, seletividade de junções. Este é um problema difícil de ser resolvido a partir de estatísticas, pois, necessitaria de uma série de histogramas que, nem sempre se encontram disponíveis. Uma estratégia para a resolução deste tipo de problema é a redistribuição dinâmica dos dados para equilibrar a execução.

A escolha do número de processadores e sua designação são mais um dos problemas deste ambiente difíceis de serem resolvidos, pois dependem do volume de dados disponíveis (casamento da taxa de produção e consumo) e de informações de disponibilidade de recursos dos nós a serem utilizados. A escolha imprecisa do grau de paralelismo a ser utilizado pode introduzir outros problemas na execução. Na execução de consultas com paralelismo intra-operador, por exemplo, a alocação excessiva pode levar a disputa de dados pelos processadores e caracteriza uma falta de ineficiência. Já o contrário poderia aumentar o tempo de execução.

Os dois últimos problemas apresentado estão intimamente relacionados ao algoritmo de escalonamento apresentados neste trabalho e, sendo assim, a capítulo seguinte explorará os trabalhos existentes nesta área.

2.2. Processamento de Consulta Adaptativo

O conceito de adaptatividade surgiu no início da década de 90 quando o processo de otimização deixou de se basear somente em informações estáticas e passou a utilizar estatísticas capturadas dinamicamente do ambiente. Esta necessidade estava inicialmente relacionada aos SGBDP que necessitavam de um mecanismo mais eficiente para designar os processadores a serem alocados na execução paralela de consultas de diferentes transações.

O processo de otimização tradicional é bastante eficiente para cenários nos quais um conjunto de informações estatísticas encontram-se disponíveis. Como por exemplo, histogramas das relações e seletividade de predicados de junções.

No final da década de 90, com a popularização da internet e a necessidade de se integrar fontes de dados nas quais estas informações estatísticas não existem, houve a necessidade de se criar técnicas mais eficientes que permitissem adaptar a execução de acordo com as variações nas características dos dados e disponibilidade de recursos dos ambientes. Neste contexto, surgem vários problemas interessantes e até então inexistentes, como: atraso na recepção dos dados e adaptação da carga de trabalho de cada site.

No contexto de processamento de *streams* surgem novos desafios para o processamento de consultas adaptativas. Neste modelo de execução as consultas normalmente possuem elevado tempo de duração e se diferem do modelo

tradicional, pois os dados são adicionados ao sistema após o registro das consultas. Neste modelo não existe informações sobre os dados e sua característica pode mudar ao longo da execução, neste sentido é importante que o sistema consiga se adaptar a estas variações e a disponibilidade de recursos no sistema. (SAMUEL MADDEN, MEHUL A.SHAH et al., 2002) (ARVIND ARASU, BRIAN BABCOCK et al., 2003) (BETH PLALE & KARSTEN SCHWAN, 2001) são trabalhos que apresentam técnicas de adaptação neste modelo de execução.

No contexto de processamento de consulta, o conceito de adaptatividade está relacionado a sistemas que são capazes de obter informações dinâmicas e adaptar o restante da execução às variações nas características dos dados e disponibilidade de recursos. (ANASTASIOS GOUNARIS, NORMAN W.PATON et al., 2004) propõem uma organização do processo de adaptação em três fases: monitoramento, análise, resposta. Na primeira fase são capturadas informações sobre a execução, na segunda verifica-se a possibilidade de melhorias e, na terceira um conjunto de alterações são realizadas para adaptar a execução as condições atuais. Adicionalmente é proposto um *framework* para a fase de monitoramento que, se baseia na extensão dos operadores tradicionais.

Várias técnicas podem ser utilizadas no contexto de processamento de consulta adaptativo, a melhor técnica a ser empregada depende do problema a ser resolvido e do ambiente em questão. Em se tratando de sistemas nos quais a adaptação é necessária uma regra geral é que o processo de otimização deve ser o menor possível, pois as técnicas tradicionais são bastante ineficientes nestes cenários. No capítulo 3 serão apresentadas algumas técnicas de processamento adaptativo.

2.3. Grades Computacionais

O termo "Grade Computacional" foi criado em meados da década de 90 para denotar uma nova proposta de infra-estrutura em computação distribuída para ciência e engenharia avançada (IAN FOSTER, CARL KESSELMAN et al., 2001). A origem do termo vem de uma analogia a rede elétrica, na qual recursos computacionais poderiam ser consumidos como energia elétrica. Assim, quando fosse necessária alguma quantidade de poder

computacional para, por exemplo, efetuar um cálculo complexo, simplesmente "plugaríamos" nosso computador pessoal em uma grade, que nos ofereceria os recursos computacionais.

2.3.1. Banco de Dados e Grades Computacionais

Até pouco tempo não haviam estudos que apontassem as contribuições da tecnologia de banco de dados para este ambiente e, sobretudo, quais as adaptações esta tecnologia deveria sofrer para atender as necessidades desta nova classe de usuários. No início desta década, com a disseminação deste ambiente, alguns estudos foram desenvolvidos nesta área. Acredita-se que a tecnologia de banco de dados, principalmente banco de dados distribuídos, tenha muito a contribuir no que diz respeito ao compartilhamento de dados em uma grade computacional.

A maioria das aplicações desenvolvidas para o ambiente de grade era baseada em arquivo. No entanto, este ambiente foi desenvolvido para suportar vários tipos de aplicações, científicas e outras, e desta forma a integração da tecnologia de banco de dados a este ambiente torna-se importante. Em (P.WATSON, 2002) é realizado um levantamento de alguns requisitos necessários a um sistema de banco de dados para o ambiente de grade. Tais requisitos incluem serviços tradicionais de banco de dados (facilidades de consulta e atualização, transação, controle de concorrência, recuperação, etc.), assim como outros mais específicos do ambiente como:

- Escalabilidade - aplicações do ambiente de grade possuem uma demanda excessiva por desempenho e capacidade de armazenamento. Existem propostas de armazenamento de peta bytes de informação a uma taxa de um Terabyte por hora (J.SHIERS, 1997).
- Tratamento de uso imprevisível - algumas das ciências que farão uso de dados na grade serão exploratórias ou movidas pela curiosidade. Isto faz com que seja difícil de prever os tipos de acessos que serão realizados.
- Acesso dirigido por metadados - a maioria das aplicações de grade utiliza metadados de maneira muito simples. Na maioria das vezes apenas para nomear nomes lógicos em arquivos físicos. Contudo, com a expansão das aplicações de grade para novas áreas, o desenvolvimento de ferramentas e

sistemas de metadados mais sofisticados será necessário. O resultado esperado é que se obtenha uma grade semântica, que seria semelhante à *semântica web* (T.BERNERS-LEE & T.J.HENDLER, 2001).

- Federação de múltiplos bancos de dados – um dos objetivos do ambiente de grade é estimular a publicação aberta de dados científicos. Se isto acontecer, é esperado que muitos dos avanços provenientes do ambiente de grade sejam de aplicações que conseguem combinar dados de várias fontes de dados distintas.

As ferramentas disponíveis para a criação de uma grade computacional (i.e. Globus (Globus) e Legion (Legion))foram desenvolvidas pensando em oferecer os serviços de infra-estrutura necessários a este ambiente como: segurança, transferência de arquivos, escalonamento e outros. A característica de *single sign on* oferecida pelo serviço de segurança do *globus*, por exemplo, pode ser um mecanismo importante na integração de banco de dados, pois com ela o usuário não necessitará informar o seu usuário e senha sempre que for acessar uma fonte de dados. Entretanto, estes serviços não são suficientes para realizar a integração da tecnologia de banco de dados ao ambiente de grade. A maioria dos serviços oferecidos é em nível de arquivo e os bancos de dados oferecem um conjunto de operações muito mais rico como, por exemplo, suporte a consulta e transações.

A tecnologia de *serviços web* vem desenvolvendo um papel importante na integração de sistemas, isolando clientes de aspectos tecnológicos próprios a cada serviço. Neste sentido o *Global Grid Forum* desenvolveu a *Open Grid Service Architecture* (OGSA) que estende a tecnologia de serviços web com novas funcionalidades como: extensão de interfaces, metadados sobre serviços, manutenção de estado entre invocações (*stateful*), notificação e gerência do ciclo de vida que permitem o desenvolvimento de aplicações mais complexas.

2.3.1.1. OGSI e OGSA

O OGSI (Open Grid Services Infrastructure) (S.TUECKE, K.CZAJKOWSKI et al., 2003) define um conjunto de especificações para a construção de serviços para o ambiente de grade. Esta especificação foi proposta pela *Global Grid Forum* (GGF) e consiste na definição de um conjunto de

interfaces WSDL para os *grid-services*. A OGSA é uma implementação de referência deste padrão.

O OGSI pode ser visto como uma extensão da tecnologia de serviços web que viabiliza a criação de aplicações mais complexas. As principais interfaces especificadas pela OGSI são:

- Descoberta: oferece informações para a localização e acesso aos serviços.
- Criação dinâmica de serviços: Define um mecanismo através do qual o usuário pode criar uma instância de um serviço do grid.
- Gerenciamento do tempo de vida: permite que os usuários de um serviço acompanhem seu estado.
- Notificação: oferece um mecanismo de comunicação assíncrona entre serviços.

As seções a seguir descrevem alguns projetos de integração de dados desenvolvidos para o ambiente de grade.

2.3.2. OGSA-DAI e OGSA-DQP

O OGSA-DAI (*Data Acces and Integration*) e OGSA-DPQ (*Distributed Query Processing*) (M.ALDPDEMIR, A.MUKHERJEE et al., 2003) são projetos de desenvolvidos pela *UK Database Task Force* cujo objetivo é produzir um sistema de integração de dados para o ambiente de grade.

O DAI é um serviço que tem por objetivo expor fontes de dados (i.e. banco de dados e arquivos xml) ao ambiente de grade. Este serviço oferece uma interface bem definida pela qual o usuário pode acessar e modificar os dados. Em especial esta interface permite a composição em um *workflow* de várias instâncias deste serviço, nos quais tarefas intermediárias de transformação dos dados podem ser realizadas por XSLT. Adicionalmente um conjunto de metadados pode ser associado a cada fonte de dados acessados futuramente.

Já o DQP é um serviço de processamento de consulta que permite a integração de várias fontes de dados DAI. Este serviço adapta várias técnicas da área de banco de dados paralelos de forma a oferecer o uso de paralelismo

implícito para aplicações de dado intensivas. O DQP pode ser dividido em dois componentes principais: GDQS (*Grid Distributed Query Service*) e o QES (*Query Evaluation Service*). O GDQS desempenha um papel de coordenador e, funciona como ponte de acesso para os clientes do serviço. Ao ser iniciado ele carrega todas as informações necessárias (i.e. metadados sobre as fonte e dados e informações sobre os recursos computacionais) para a compilação, otimização, particionamento e escalonamento de consultas distribuídas sobre nós de execução da grade. Este componente foi desenvolvido a partir do encapsulamento das funções de compilação e otimização propostas em (J.SMITH, ANASTASIOS GOUNARIS et al., 2002). Em especial este componente expõe a mesma interface do DAI e permite que suas instâncias também sejam compostas em *workflow*. O QES é responsável por realizar a avaliação de fragmentos do plano de execução atribuídos pelo coordenador.

2.3.3. CoDIMS-G

O CoDIMS-G é uma instância do *framework* CoDIMS (VINÍCIUS FONTES, MARCIO L DUTRA et al., 2004) (FABIO PORTO, VINÍCIUS FONTES et al., 2005) criada para suportar acesso a fontes de dados e programas disponíveis no ambiente de grade.

O framework codims permite que componentes de um sistema de processamento de consulta tais como gerente de metadados, as fontes de dados, a álgebra, a linguagem de consulta e o modelo de execução de consultas sejam especificados pelo usuário de modo a se adequar a cada aplicação. Como exemplo, codims foi estendido em (A.V.LOPES, 2004) para suportar consultas xml relacional e em (FABIO COUTINHO & FABIO PORTO, 2004) como base para processamento de consultas sobre um modelo de dados semântico.

Codims utiliza um modelo de sistema do tipo mediador apesar do aspecto de integração não ter sido plenamente explorado, em particular, o sistema não possui um componente de re-escrita de consultas e resume-se a traduzir via adaptadores os dados das fontes para o modelo de dados adotados.

Os principais componentes da arquitetura do CoDIMS-G são ilustrados pela figura 3. O gerente de metadados possui metadados sobre as fontes de dados (esquema, informações estatísticas (cardinalidade)) e o ambiente de execução,

como por exemplo, nós disponíveis, sua vazão, softwares disponíveis, memória, espaço em disco.

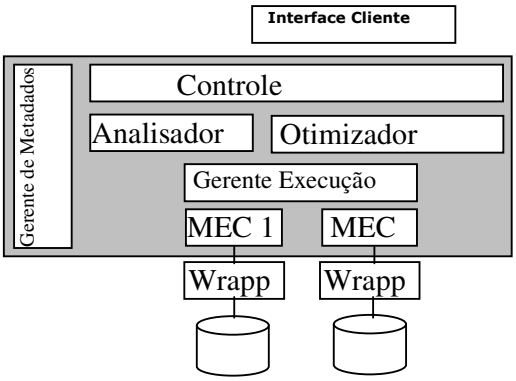

Figura 3. Arquitetura do CoDIMS-G.

Nesta arquitetura, um serviço executado em um nó *gatekeeper* realiza o papel de comunicação com o mundo externo à grade. As requisições realizadas pelos usuários são encaminhadas ao componente de controle que é responsável por determinar o conjunto de ações necessárias para atender o usuário. Em especial, as consultas submetidas são encaminhadas para o subsistema de processamento de consulta (SSPC) que é composto por um Analisador, um Otimizador, um Gerente de Execução e uma Máquina de Execução de Consulta (MEC).

No SSPC a execução tem início no Analisador que criará uma representação de grafo para o plano de execução de consulta (PEC) e realizará a análise sintática e semântica da consulta segundo o esquema das fontes de dados definidas no catálogo. Após esta etapa, a consulta é então enviada ao otimizador que definirá a melhor estratégia de execução e criará um plano físico de execução (PFE). Sempre que a execução paralela de um operador ou fragmento do plano for possível o Escalonador (VINÍCIUS FONTES, MARCIO L DUTRA et al., 2004) (FABIO PORTO, VINÍCIUS FONTES et al., 2005) será acionado para ajudar na escolha do conjunto de nós a ser utilizado. Concluído o processo de otimização o PFE é encaminhado ao Gerente de Execução que tem a responsabilidade de instanciar a MEC nos nós indicados e garantir que o processamento seja realizado com sucesso. Cada instância da MEC receberá o fragmento do plano que deverá executar e é responsável por instanciar os operadores algébricos e de controle

necessários, incluindo os operadores de comunicação. O acesso às fontes de dados é realizado via *Wrappers*, que têm a função de converter os dados do modelo local para o modelo global de execução.

2.4.Síntese do Capítulo

Neste capítulo abordamos conceitos de base para o bom acompanhamento deste trabalho. Os assuntos discutidos sugerem os aspectos mais relevantes e para os quais o leitor deve se preparar de modo a seguir agradavelmente o restante dessa dissertação. Apresentam-se, desta forma, aspectos principais de bancos de dados paralelos e de processamento de consultas neste ambiente. Discutem-se conceitos associados às técnicas de adaptatividade de consulta e, por fim, o ambiente de grade como hospedeiro das aplicações e serviços aqui discutidos.