Uma Arquitetura para o Processamento de Consultas com Agregados

Nilton Cézar de Paula[1], José Craveiro da Costa Neto[2], Liria Matsumoto Sato[3]
[1]*Departamento de Computação, Universidade Estadual de Mato Grosso do Sul*
Dourados-MS, Brasil
ncdepaula@dourados.br
[2]*Departamento de Computação e Estatística, Universidade Federal de Mato Grosso do Sul*
Campo Grande-MS, Brasil
craveiro@dct.ufms.br
[3]*Departamento de Engenharia de Computação e Sistemas Digitais, Escola Politécnica da USP*
São Paulo-SP, Brasil
liria@pcs.usp.br

Resumo

O processamento paralelo e distribuído é uma alternativa para melhorar o desempenho de consultas sobre um data warehouse. Ultimamente, vem-se dando ênfase ao uso da técnica de agrupamento prévio no processamento de consultas com agregados em sistemas centralizados. A sua aplicação permite maior eficiência no processamento da consulta, com a execução do agrupamento antes da junção. Neste trabalho, propõe-se uma arquitetura de software para a execução de consultas com agregados explorando o paralelismo e o agrupamento prévio. Essa arquitetura traz novos recursos ao CDBS (Concurrent Database System) e viabiliza um ambiente para a análise de consultas com agregados. Os resultados mostram que a aplicação do agrupamento prévio num ambiente paralelo pode diminuir o tempo de execução da consulta e os acessos a disco. Porém, essa técnica deve ser evitada quando o número de grupos gerados por uma consulta é muito grande. Para tanto, deve-se identificar quando essas situações ocorrem utilizando-se algum critério, por exemplo, estimar custos.

1. Introdução

Consultas em SQL com agregados têm sido comuns em bancos de dados para suporte à decisão, como os *data warehouses*. Essas consultas são utilizadas geralmente para auxiliar a tomada de decisões nas empresas e são denominadas consultas analíticas. Consultas com agregados particionam os dados em grupos e aplicam funções de agregação (por exemplo, COUNT, SUM, AVG, MAX e MIN) sobre cada grupo. Os dados podem ser derivados de diferentes tabelas usando a operação de junção. Devido ao grande volume de informações existentes nos *data warehouses*, é indispensável a aplicação de técnicas para processar as consultas de maneira eficiente [1].

O processamento paralelo tem sido uma opção interessante para melhorar o desempenho na execução de consultas em *data warehouses*, pois os dados distribuídos podem ser processados por diversos processadores simultaneamente. Assim, o paralelismo aplicado nesses bancos de dados oferece oportunidades para melhorar o desempenho no processamento de consultas. No entanto, poucos estudos têm tratado a questão do processamento paralelo em consultas com agregados [1,2].

Costa Neto apresenta em [3] uma proposta do sistema CDBS (*Concurrent Database System*), cujo principal objetivo é oferecer alto desempenho em consultas analíticas. O CDBS é constituído por uma biblioteca de funções para a criação, consulta e manutenção de um banco de dados de produção e um *data warehouse* paralelos, integrados num único ambiente.

Pode-se melhorar o desempenho do processamento de consultas com agregados com a execução do operador *group-by* antes do operador de junção. A esta técnica dá-se o nome de agrupamento prévio. Com a sua aplicação, consegue-se reduzir o custo de execução de operações de junção [4,5].

Neste artigo, registra-se o esforço de pesquisa desenvolvido para apresentar uma arquitetura de software para a execução de consultas com agregados usando o processamento paralelo e a aplicação do agrupamento prévio. Com a implementação da arquitetura aqui proposta usando a LAM-MPI [6], busca-se incluir uma nova facilidade no CDBS, oferecendo mais uma solução com bom desempenho para o ambiente.

O restante do artigo está organizado em 6 seções. A seção 2 apresenta uma discussão dos trabalhos

relacionados com o assunto desta pesquisa. A seção 3 mostra os detalhes relacionados com o agrupamento prévio. A seção 4 apresenta a arquitetura proposta. Já a seção 5 apresenta alguns resultados obtidos com a utilização da arquitetura proposta com a execução de um estudo de caso. Finalmente, a seção 6 apresenta as conclusões e contribuições do trabalho.

2. Trabalhos correlatos

Yan e Larson apresentam em [5] um estudo para a utilização do agrupamento prévio. Apresentam a transformação de uma consulta em SQL em outras duas: uma com o operador *group-by* e a outra sem esse operador. O resultado da consulta é determinado pela junção dos resultados obtidos com a execução das duas consultas. Devido ao fato de estar tratando duas consultas separadamente, a ordem das junções fica restrita, reduzindo o espaço de busca pelo otimizador.

Um outro estudo, apresentado por Chaudhuri e Shim [4], trata a questão transformando um plano de consulta já otimizado em um outro equivalente. Na construção do plano equivalente, a ordem das junções e dos agrupamentos são considerados, diminuindo desta forma, o problema do espaço de busca pelo otimizador.

Em [7] é apresentada uma estratégia para o processamento paralelo de agregados, a qual foi implementada no projeto Gamma [8]. Sua abordagem tradicional é baseada num único coordenador central. O resultado do processamento em cada nó é enviado para o coordenador central, que compõe o resultado final. Essa abordagem é indicada para consultas em que o número de grupos gerados é pequeno.

O desempenho de consultas analíticas é um fator crítico, portanto, a idéia deste artigo é aplicar o agrupamento prévio e o paralelismo para melhorar o desempenho no processamento.

3. Agrupamento prévio

A seguir, apresentam-se os tipos de transformação, as definições e as condições necessárias para a aplicação da técnica de agrupamento prévio segundo os estudos de Chaudhuri e Shim [4].

3.1. Tipos de transformação

Existem três tipos de transformação para o agrupamento prévio: invariante, de união simples e de união generalizada.

No **agrupamento invariante** o operador *group-by* é movimentado da raiz para algum nó interno do plano de consulta.

No **agrupamento de união simples**, novos operadores *group-by* são adicionados ao plano de consulta, correspondendo a uma generalização do **agrupamento invariante**. Os operadores *group-by* adicionais são necessários tendo em vista que as operações subseqüentes poderão gerar mais de uma tupla por grupo. Por esta razão, essas tuplas precisam ser juntadas ao mesmo grupo. A união é possível porque as funções de agregação podem ser calculadas em partições.

O **agrupamento de união generalizada** é um tipo de **agrupamento de união simples**; a diferença é que para cada operador *group-by* adicionado no plano, também é adicionado um contador de tuplas por grupo. Para aplicar esta transformação, as funções de agregação devem ser calculadas seguindo as seguintes derivações: MIN(N,s)=s, MAX(N,s)=s, AVG(N,s)=s, SUM(N,s)=N*s, e COUNT (N,s)=N, onde **N** é o número de tuplas **s** em um grupo.

Cada transformação apresentada está numa ordem de generalidade e complexidade crescentes. As transformações adicionam complexidade para a construção do plano de execução de uma consulta em relação aos planos tradicionais, pois o espaço de busca por uma solução é muito maior. Com isto, alguns problemas surgem: 1) A quantidade de planos de execução equivalentes cresce rapidamente quando se considera a ordem de interesse das junções e dos agrupamentos; 2) É difícil construir um plano que satisfaça, a um mesmo tempo, a ordem de interesse das junções e dos agrupamentos e 3) Se os operadores *group-by* são implementados usando uma técnica de ordenação, o número de ordens de interesse cresce.

Essas transformações devem ser aplicadas levando em consideração um modelo de estimativa de custos para melhor orientar a sua aplicação. No entanto, deve-se também observar os problemas apresentados para que a enumeração de planos não seja proibitiva.

3.2 Conjuntos necessários para a transformação

Para aplicar uma das transformações é necessária a apresentação de algumas definições.

Por exemplo, numa consulta SQL com agregados, analisam-se seus componentes (colunas de agrupamento e de agregação) e anotam-se informações no plano de consulta. Essas anotações são úteis para determinar o tipo de agrupamento prévio a empregar.

Em cada nó do plano de consulta, são definidos três conjuntos: as colunas de junção, as colunas requeridas e as colunas candidatas à agregação. As **colunas de junção** são as colunas de um nó do plano que participam em predicados de junção que são avaliados por nós ancestrais. As **colunas requeridas** de um nó do plano compreendem as suas colunas de junção e as colunas de agrupamento da consulta. As **colunas candidatas à agregação** são as colunas de um nó do plano que também são colunas de agregação da consulta, mas não estão entre as colunas requeridas.

A definição de colunas candidatas à agregação exclui as colunas de agregação que participarão de predicados futuros, embora sejam colunas requeridas, devendo ser preservadas no plano por um nó até que os predicados sejam avaliados.

As colunas de um determinado nó do plano que deverão ser preservadas para um processamento subseqüente estarão entre as colunas requeridas e as colunas candidatas à agregação. Desta forma, essas colunas deverão ser retidas por uma operação de projeção. Essas colunas também determinam a configuração dos novos operadores *group-by* para um determinado nó do plano de consulta da seguinte maneira: 1) O conjunto de colunas de agrupamento da consulta é o conjunto de colunas requeridas; e 2) O conjunto de colunas de agregação da consulta é o conjunto de colunas candidatas à agregação.

3.3. Identificando o nó de transformação

O nó *n* de um plano de consulta que será escolhido para a aplicação do agrupamento prévio deverá satisfazer uma ou mais condições dentre as apresentadas a seguir:

1- Cada coluna de agregação de uma consulta está no conjunto de colunas candidatas à agregação de *n*;

2- Cada coluna de junção de *n* é também uma coluna de agrupamento de *n* e

3- Os nós ancestrais a *n* avaliam cada predicado de junção sobre uma chave estrangeira que é também coluna de *n*.

A primeira condição não permite que o nó candidato realize a agregação sobre colunas que pertencem a outras tabelas, pois se assim fosse, tuplas poderiam ser perdidas na agregação. Neste caso, esta condição será satisfeita assim que uma operação de junção determine a união das colunas de agregação da consulta avaliada.

As outras condições determinam o nó apropriado de tal forma que as junções subseqüentes não resultem em mais que uma tupla para um mesmo grupo através de chaves estrangeiras.

Um nó apresenta a propriedade de **agrupamento invariante** quando ele satisfaz as três condições acima. Também terão esta propriedade seus nós ancestrais e qualquer um deles poderá se escolhido para a movimentação do operador *group-by*.

No **agrupamento de união simples**, não é obrigatório que as colunas de junção sejam iguais às correspondentes colunas de agrupamento da consulta, dispensando a segunda condição. Desta forma, o nó escolhido só precisa satisfazer as condições 1 e 3.

Um nó será identificado com a propriedade de **agrupamento de união generalizada** quando algumas colunas de agregação da consulta estão presentes neste nó, satisfazendo de forma parcial a condição 1.

Com o resultado desta discussão fica fácil deduzir que a aplicação da técnica de agrupamento prévio deve reduzir significativamente o tempo de execução de uma operação de junção. Como conseqüência disto, o tempo de execução de uma consulta deverá ser melhor.

4. Arquitetura para execução de consultas usando processamento paralelo

Apresenta-se aqui a arquitetura de software proposta para a execução de consultas com agregados num ambiente de memória distribuída [9], buscando identificar nas consultas os comportamentos de sua execução quando aplicada a técnica de agrupamento prévio.

A arquitetura possui dois componentes básicos: um **módulo cliente**, que recebe uma consulta escrita em SQL, analisa-a, constrói seu plano de execução eficiente, controla sua execução e devolve os resultados ao usuário; e um **módulo servidor de arquivos paralelos**, que armazena e acessa os dados nos discos e atende às solicitações dos clientes. Na Figura 1, tem-se uma visão geral da arquitetura proposta. A seguir, faz-se uma apresentação detalhada dos módulos desta arquitetura.

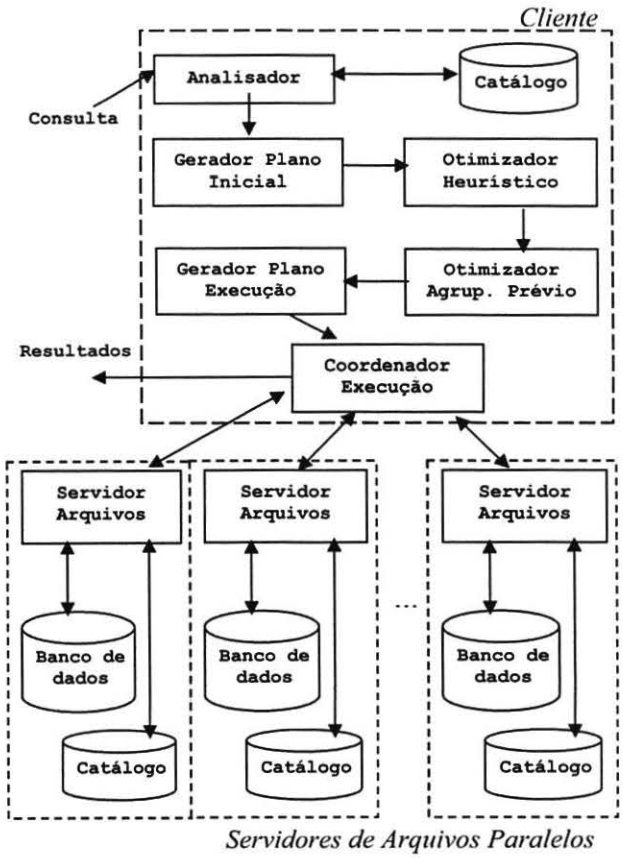

Figura 1. Arquitetura para a execução de consultas no CDBS

4.1. Módulo cliente

O módulo cliente, como responsável por preparar uma consulta para ser executada, possui os seguintes componentes: analisador, gerador de plano inicial, otimizador heurístico, otimizador de agrupamento prévio, gerador de plano de execução, coordenador de execução e o catálogo de informações da base de dados.

O **analisador** recebe uma consulta escrita em SQL e faz uma série de análises, para garantir sua execução corretamente. Para tanto, verifica se as tabelas e atributos estão presentes no catálogo do banco de dados e as condições estão completamente escritas. Constrói, também algumas estruturas auxiliares (por exemplo, a lista de *tokens*) para serem utilizadas pelos demais componentes.

O **gerador do plano inicial** transforma a consulta numa representação não otimizada usando uma árvore de profundidade à esquerda conforme [10]. Esta árvore é o ponto de partida para serem aplicadas tanto a otimização heurística como também a otimização de agrupamento prévio.

O **otimizador heurístico** aplica regras heurísticas usando a álgebra relacional para construir um plano de consulta melhorado [10]. As tarefas desenvolvidas por este componente são essenciais porque o plano construído estará preparado para ser transformado, se possível, pelo próximo componente, o otimizador de agrupamento prévio.

O **otimizador de agrupamento prévio** verifica a possibilidade da aplicação da técnica de agrupamento prévio no plano de consulta já otimizado pelo componente anterior. Para isto, são identificados no plano os nós que possuem propriedades de agrupamento (invariante, de união simples ou de união generalizada), sendo aplicada a nova otimização.

O **gerador do plano de execução** é responsável por determinar os caminhos de acesso e os métodos de implementação para cada operação do plano de consulta.

O **coordenador de execução** distribui o plano de execução da consulta disparando os processos para os servidores de arquivos que cooperarão em sua execução. A decisão em escolher um servidor é tomada levando em consideração a disponibilidade do servidor e a forma (horizontal ou vertical) e o tipo de distribuição (circular, *hash* ou faixa de valores) dos dados [11,12]. Será responsável, também pela composição do resultado final da consulta.

O **catálogo** armazena informações que serão utilizadas pelos componentes da arquitetura, por exemplo, a descrição das tabelas e seus atributos e a política de replicação e fragmentação dos dados. O acesso a um catálogo é um fator crítico, pois seus dados deverão ser requisitados várias vezes; por isto, recomenda-se utilizar uma estrutura de dados que consiga oferecer um bom desempenho no acesso aos dados, por exemplo, uma tabela *hash* [13]. No entanto, num ambiente paralelo, deve-se adotar outras estratégias para garantir o acesso eficiente a um catálogo. Na arquitetura proposta, o catálogo é replicado em cada servidor de arquivos paralelos para oferecer essa eficiência.

4.2. Módulo servidor de arquivos paralelos

O módulo servidor de arquivos paralelos é responsável por armazenar e acessar os dados em disco. O acesso é feito através do conhecimento de como os dados estão armazenados no banco de dados; para isto utiliza informações existentes no catálogo. Na execução de uma consulta, o módulo servidor apenas cumpre as tarefas definidas no plano de execução da consulta, assim nenhuma decisão é tomada nesse módulo.

O número de módulos servidores de arquivos é determinado pelo projetista do sistema através de uma análise crítica do desempenho atual da aplicação. Caso alterações futuras na quantidade de servidores sejam identificadas, será necessário redistribuir os dados aos novos servidores. Isto também poderá ocorrer se a política de fragmentação e replicação dos dados for alterada.

Como os dados estão distribuídos pelos nós da rede, a arquitetura proposta fornece uma excelente oportunidade para otimizar o processamento das consultas analíticas submetidas ao CDBS. O paralelismo dos dados e dos processos pode, por exemplo, ser usado para exploração da localidade dos acessos.

5. Um estudo de caso: uma rede de locadoras de filmes

Para ilustrar a viabilidade da arquitetura proposta, faz-se, nesta seção, a apresentação de um estudo de caso. O problema escolhido é um sistema para uma rede de locadoras de filmes para vídeo-cassete [3].

As lojas da rede estão distribuídas por vários bairros de uma grande cidade e recebem remuneração pelos serviços prestados de aluguel de cópias de filmes a seus clientes. Além da rotina cotidiana das lojas da rede, a diretoria e as gerências das lojas necessitam retirar do sistema uma série de informações que possam servir de auxílio na tomada de decisões.

5.1. Descrição do banco de dados

O domínio do problema está relacionado a situações reais no âmbito de organizações que comercializam seus produtos e serviços, pois elas trabalham com filiais que estão geograficamente distribuídas e precisam atender às suas coordenações e direções centrais. As tabelas a seguir

representam o banco de dados da rede de locadoras de maneira resumida.

CLIENTE (*idcli, nome, idloja, rua, bairro, cep, fone*)
LOJA (*idloja, nome, rua, bairro, cep, zona, fone*)
FILME (*idfilme, nome, genero, censura, produtora*)
LOCACAO (*idloc, idcliloc, idlojaloc, idfilmeloc, dia, mes, ano, vrloc*)

As três primeiras tabelas disponibilizam informações à tabela LOCACAO, que ocupa posição privilegiada no sentido de oferecer informações para a tomada de decisões.

Para garantir um bom desempenho nas consultas, maximizando o paralelismo, explorou-se a localidade dos dados através de políticas de replicação e fragmentação. Assim, os dados utilizados nos experimentos foram armazenados nos servidores de arquivos paralelos utilizando-se os passos do algoritmo a seguir no módulo cliente:

1- Obtenha as informações do catálogo e determine a quantidade de servidores de arquivos paralelos e inicie em cada um o módulo servidor;

2- Replique os dados gerados das tabelas LOJA e FILME aos servidores;

3- A partir do número de lojas, determine a política de fragmentação para as tabelas CLIENTE e LOCACAO determinando a faixa de valores correspondente a cada servidor;

4- Distribua os dados gerados das tabelas CLIENTE através do atributo *idloja* e LOCACAO usando o atributo *idlojaloc*, de acordo com a política definida no passo 3.

A partir da execução desses passos o banco de dados estará pronto para ser utilizado durante o processamento das consultas.

5.2. Execução do estudo de caso

Para execução do estudo de caso foram definidas duas consultas, exemplificadas como Consulta 1 e Consulta 2 que possuem a propriedade de agrupamento de união simples. A diferença existente entre essas consultas é que a primeira resulta num pequeno número de grupos, enquanto a segunda gera um número maior de grupos durante os agrupamentos.

Seja a consulta 1: definir a quantidade de fitas locadas e o valor faturado com locações, por bairro, anualmente, para filmes do gênero 'drama' e locações efetuadas a partir de 1993. A consulta em SQL pode ser representada como a seguir:

```
SELECT bairro, ano, COUNT(*), SUM(vrloc)
FROM Loja, Locacao, Filme
WHERE idloja=idlojaloc AND idfilme=idfilmeloc
        AND ano>=1993 AND genero='drama'
GROUP BY bairro, ano
```

Para analisar a viabilidade técnica desta proposta faz-se a comparação dos resultados usando a otimização tradicional e a otimização com o agrupamento prévio.

Na otimização tradicional, a consulta 1 executa a junção de LOJA, LOCACAO e FILME, particiona os dados em grupos (*bairro* e *ano*) e, por fim, aplica as funções de agregação (*COUNT(*)* e *SUM(vrloc)*) a cada grupo. Essa situação está representada na Figura 2, que apresenta apenas as tabelas e operações de junção e agregação para simplificar a análise das informações do plano de consulta.

O operador γ, utilizado no plano de consulta, representa a operação de agrupamento de agregação de consulta, conforme [14].

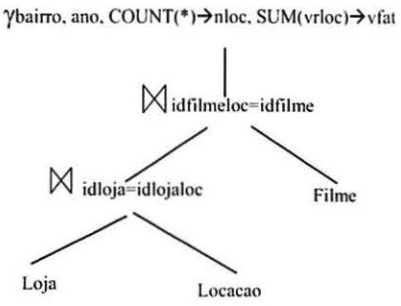

Figura 2. Plano de consulta tradicional simplificado

Outro plano de consulta para a execução da consulta 1 está mostrado na Figura 3 que também apresenta apenas as tabelas e operações de junção e agregação, correspondendo aos seguintes passos:

1- Particionar os dados em grupos (*idlojaloc*, *idfilmeloc* e *ano*) e para cada grupo aplicar as funções de agregação (*COUNT(*)* e *SUM(vrloc)*);

2- Aplicar a junção de LOJA com o resultado do passo anterior;

3- Particionar os dados que foram gerados pelo passo anterior em grupos (*idfilmeloc*, *bairro* e *ano*) e, para cada grupo aplicar as funções de agregação (*SUM(nloc)* e *SUM(vfat)*);

4- Aplicar a junção de FILME com o resultado do passo anterior e

5- Particionar os dados em grupos (*bairro* e *ano*) e para cada grupo, aplicar as funções de agregação (*SUM(nloc)* e *SUM(vfat)*), compondo o resultado final.

O agrupamento do passo 1 define a quantidade de locações por ano para cada filme e loja e o faturamento correspondente. Observa-se que há necessidade de preservar alguns atributos da tabela LOCACAO (*idlojaloc*, *idfilmeloc*), pois eles serão utilizados em operações posteriores. No segundo agrupamento, obtém-se o resultado da junção entre o resultado do passo 1 com a tabela LOJA. O último agrupamento define a

composição do resultado final, visto que os dados estão particionados pelos atributos *idfilmeloc*, *bairro* e *ano*.

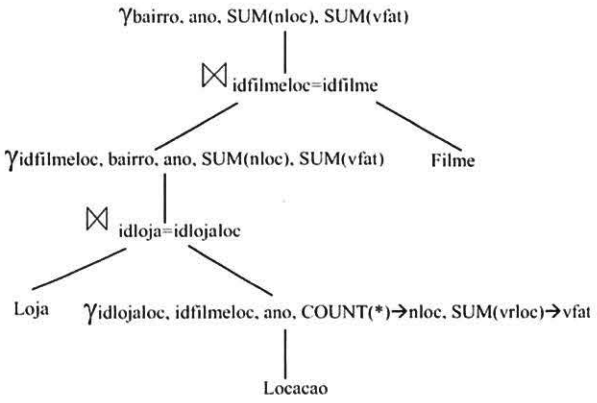

Figura 3. Plano de consulta simplificado com agrupamento prévio

O plano de consulta com agrupamento prévio é construído a partir da identificação dos três conjuntos, conforme apresentado na seção 3.2, no plano tradicional, como por exemplo, para o nó LOCACAO: colunas de junção (*idlojaloc*, *idfilmeloc*), colunas requeridas (*idlojaloc*, *idfilmeloc*, *ano*), e colunas candidatas à agregação (*, *vrloc*). Assim, a configuração para o nó LOCACAO será determinada pelas colunas dos últimos dois conjuntos (γidlojaloc, idfilmeloc, ano, COUNT (*)→nloc, SUM (vrloc)→vfat) e está representada na Figura 3.

Foi escolhida mais uma consulta para a execução do estudo de caso, a consulta:

```
SELECT bairro, ano, mes, dia, COUNT(*), SUM(vrloc)
FROM Loja, Locacao, Filme
WHERE idloja=idlojaloc AND idfilme=idfilmeloc
      AND ano>=1993 AND genero='drama'
GROUP BY bairro, ano, mes, dia
```

Assim, pode-se identificar em que circunstância é viável a aplicação da técnica de agrupamento prévio.

5.3. Resultados experimentais

Nesta seção, apresentamos os resultados obtidos com os experimentos realizados com a implementação dos componentes da arquitetura proposta.

Os experimentos foram realizados utilizando-se 1, 2, 4, 8 e 16 servidores de arquivos paralelos conectados a uma rede Ethernet de 10Mbps. Cada servidor com um único processador Pentium III 1.3GHz, com 256 MB de memória e 2GB de disco com o sistema Linux e o ambiente LAM-MPI.

Para as análises comparativas foram utilizados os resultados das anotações das medidas de tempo de execução das consultas 1 e 2.

Inicialmente, são criadas as tabelas nos diversos servidores de arquivos paralelos a partir das informações existentes no catálogo do banco de dados. Depois, os dados são inseridos nas tabelas usando uma rotina para a geração randômica dos dados. Algumas características deste banco de dados são mostradas na Tabela 1. Finalizadas essas duas etapas, a criação e a geração, o banco de dados está pronto para a execução de consultas.

Tabela 1. Características do banco de dados

Tabelas	Nº de Registros	Nº de atributos	Comprimento do registro (bytes)	Tamanho (Kb)
Cliente	1024	10	93	93,0
Loja	16	12	121	1,9
Filme	64	9	77	4,8
Locação	1.440.000	9	44	61875,0

O primeiro experimento foi realizado executando a consulta 1. Assim, os agrupamentos que são realizados durante a execução da consulta, como mostrado na Figura 3, possibilitarão a análise do comportamento do tempo de execução da consulta e do espaço necessário para sua execução.

A Figura 4 apresenta o tempo de execução da consulta 1 com até 16 servidores de arquivos paralelos nas duas modalidades: sem o uso da técnica de agrupamento prévio (SAP) e com o uso da técnica de agrupamento prévio (CAP). Com a aplicação da técnica CAP, obteve-se em todas configurações de servidores o melhor desempenho em relação à técnica SAP. Assim, as consultas que particionam os dados em poucos grupos são beneficiadas com esta técnica. O gráfico também mostra que á medida que se aumenta o número de servidores, o tempo de execução da consulta diminui, isto porque os dados distribuídos permitem a aplicação do paralelismo. Cada servidor contribui para a execução da consulta levando em consideração a localidade dos dados.

Um fato interessante a ser observado no gráfico da Figura 4 é que a diferença entre as técnicas SAP e CAP torna-se menor à medida que se aumenta o número de servidores de arquivos paralelos. Este fato ocorre devido à sobrecarga existente para compor o resultado final no cliente e a amostra de dados não é muito grande, mas mesmo assim a técnica CAP é três vezes mais rápida que a SAP com 16 servidores. A Figura 5 apresenta esse comportamento com mais clareza.

Na consulta 2, o tempo de execução da consulta com a técnica CAP foi superior em relação à técnica SAP, isto indica que o agrupamento prévio neste tipo de consulta deve ser evitado porque essa consulta particiona os dados

numa grande quantidade de grupos. A Figura 6 mostra o comportamento dessa consulta.

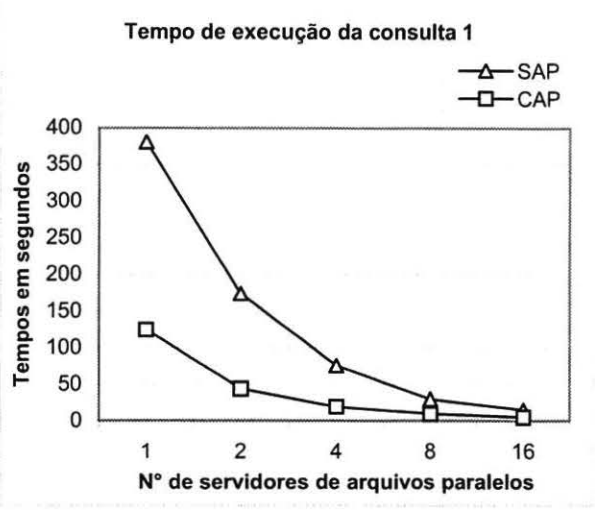

Figura 4. Comparativo de tempo de execução da consulta 1

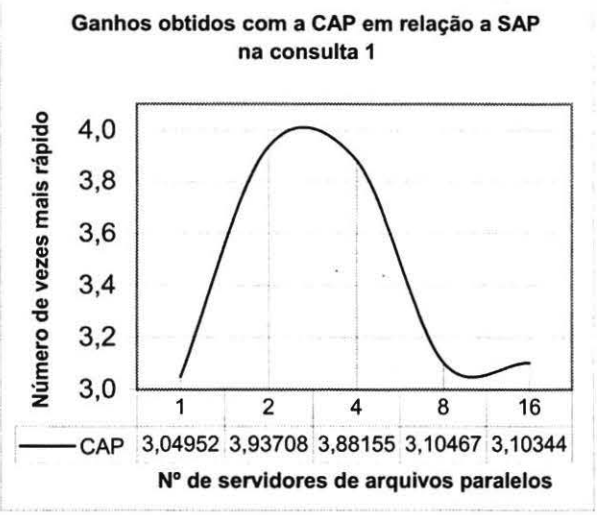

Figura 5. Ganhos em velocidade com a técnica CAP na consulta 1

A técnica de agrupamento prévio também se revelou interessante quanto à utilização do espaço em disco para o armazenamento temporário dos resultados de cada operação da consulta 1. Apesar da aplicação de três agrupamentos, como mostrado na Figura 3, o acesso a disco não influenciou negativamente no tempo de execução da consulta. O espaço em disco requerido com a técnica CAP foi de 4,7 vezes menos em relação à técnica SAP, um fator de grande importância porque o acesso a disco ainda é o gargalo em banco de dados. No entanto, esse mesmo comportamento não ocorreu com a consulta 2, contribuindo ainda mais para que o agrupamento prévio não seja aplicado nesse tipo de consulta. A Figura 7 apresenta o espaço requerido pelas consultas.

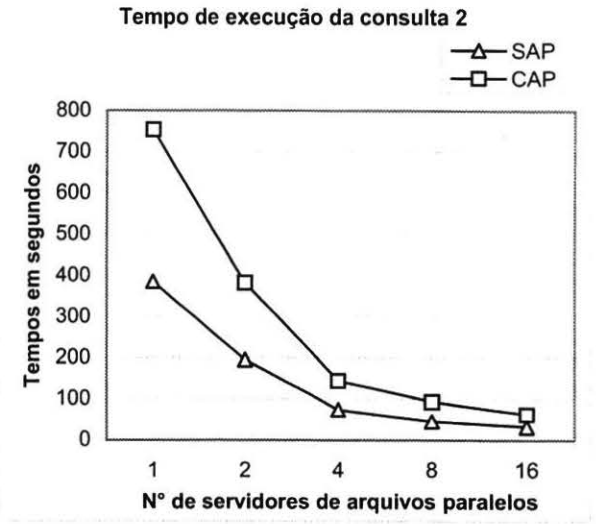

Figura 6. Comparativo de tempo de execução da consulta 2

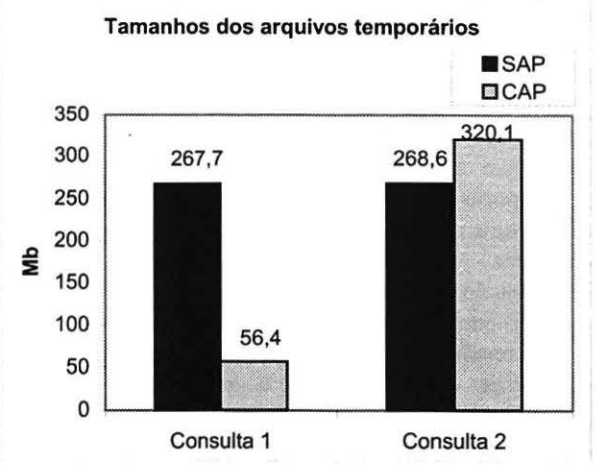

Figura 7. Espaço em disco dos arquivos temporários

Os custos dominantes para a execução das consultas são apresentados no gráfico da Figura 8. Quando a técnica SAP é aplicada, as duas consultas têm seu custo dominante pela operação de junção. Porém, quando a técnica CAP é aplicada na consulta 1, o custo dominante passa ser o da operação de agrupamento, mas em proporções menores em relação á técnica SAP. O baixo desempenho da consulta 2 com a técnica CAP deve-se ao fato de se gerar uma grande quantidade de grupos, o que não gerará diminuição significativa no custo da operação de agrupamento e, muito menos, na operação de junção.

O uso da técnica CAP revela-se bastante viável em ambientes paralelos de grande porte quando se trata da execução de consultas analíticas, principalmente, porque

este tipo de consulta usa a operação de agrupamento com bastante freqüência, com redução significativa do número de grupos.

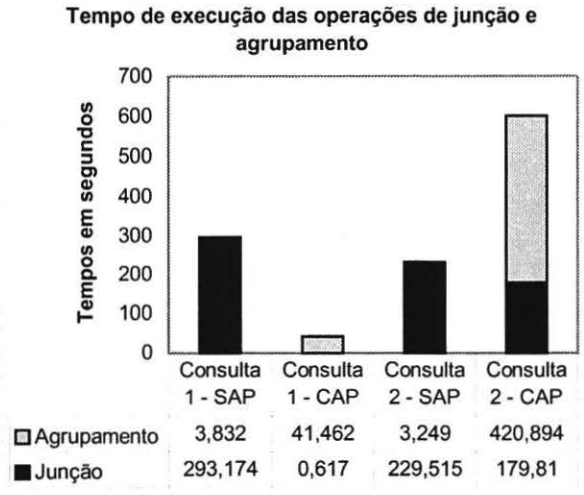

Figura 8. Comparativo de tempo de execução das operações de junção e agrupamento

6. Conclusões e trabalhos futuros

A aplicação da técnica de agrupamento prévio permite ganho de desempenho em consultas com agregados submetidas a grandes quantidades de informação. Identificar o comportamento dessas consultas constitui um importante aspecto para a aplicação desta técnica. A arquitetura proposta e implementada para a execução de consultas em um ambiente paralelo permitiu a viabilização dos experimentos para a descoberta de comportamentos.

Os resultados experimentais obtidos através de um estudo de caso revelaram a importância do uso do paralelismo nas consultas analíticas. Aproveitando-se da localidade dos dados, o processamento realizado em cada módulo servidor de arquivos paralelos mostrou-se eficaz.

Outro importante aspecto quanto à utilização do agrupamento prévio é a determinação de seu uso ou não no processamento da consulta. Esta decisão pode ser tomada levando em consideração um modelo de estimativas de custos que auxiliará na descoberta do número de grupos que serão gerados quando aplicado o operador *group-by*.

Como trabalhos futuros, pretende-se definir os métodos de acesso mais apropriados, determinar um modelo de estimativas de custos que defina quando aplicar a técnica de agrupamento prévio, explorar consultas com novas características, por exemplo, com propriedade invariante e união generalizada, explorar outras técnicas de paralelismo para o processamento de agrupamentos, e construir uma interface mais amigável para a execução das consultas.

7. Referências

[1] Chaudhuri, S.; Dayal, U. An Overview of Data Warehousing and OLAP Technology. SIGMOD Record, março/1997, pp. 65-74.

[2] Garcia-Molina, H.; Labio, W. J.; Wiener, L. J.; Zhuge, Y. Distributed and Parallel Computing Issues in Data Warehousing, 1999.

[3] Costa Neto, J. C. Considerações sobre a Integração de um Banco de Dados e um Data Warehouse sobre um Sistema de Arquivos Paralelos. Tese de doutorado, Escola Politécnica da Universidade de São Paulo, 2001.

[4] Chaudhuri, S.; Shim, K. Including Group-By in Query Optimization. Proceedings of 20th International Conference on VLDB, Santiago de Chile, Chile, setembro/1994, pp. 54-366.

[5] Yan, W. P.; Larson, P. Performing Group-By before Join, Proceedings of the IEEE, 1994, pp.89-100.

[6] Wilkinson, B.; Allen, M. Parallel Programming: Techniques and Applications Using Networked Workstations and Parallel Computers. Prentice-Hall, Inc., 1999.

[7] Bitton, D.; Boral, H.; DeWitt, D. J.; Wilkinson, W. K. Parallel Algorithms for the Execution of Relational Database Operations, ACM Transactions on Database Systems, setembro/1983, pp. 324-353.

[8] DeWitt, D. J.; Ghandeharizadeh, S.; Schneider, D.; Bricker, A.; Hsiao, H.; Rasmussen, R. The Gamma Database Machine Project. IEEE Trans. On KDE, março/1990, pp. 44-62.

[9] Waqar, H.; Florescu, D.; Valduriez, P. Open Issues in Parallel Query Optimization, SIGMOD Record, setembro/1996, pp. 28-33.

[10] Elmasri, R.; Navathe, S. R. Fundamentals of Database Systems. Addison Wesley, 3 ed., 2000.

[11] DeWitt, D. J.; Gray, J. Parallel Database Systems: The Future of High Performance Database Processing. CACM, junho/1992, pp. 85-98.

[12] Yu, C. T.; Meng, W. Principles of Database Query Processing for Advanced Applications. Morgan Kaufmann Publishers, Inc., 1998.

[13] Szwarcfiter, J. L.; Markenzon, L. Estruturas de Dados e Seus Algoritmos. Segunda Edição. Editora LTC, 1994.

[14] Garcia-Molina, H.; Ullman, J. D.; Widom, J. Database System Implementation.Prentice-Hall, Inc., 2000.