Uma experiência na implementação de operadores da álgebra relacional no computador paralelo NCP I

Marta L. Queirós Mattoso
Claudio Luis de Amorim
Programa de Engenharia de Sistemas e Computação
COPPE/UFRJ

Este trabalho descreve a experiência obtida com a implementação de um protótipo de sistema de banco de dados paralelo, o PARBASE sobre a máquina paralela desenvolvida na COPPE, o NCP I, utilizando um ambiente de programação paralela. São levantadas as diversas características que exploram o paralelismo num sistema de banco de dados. A experiência da implementação do PARBASE é comentada através dos operadores da álgebra relacional de seleção e junção. Foram obtidas diversas medidas de desempenho onde verificou-se um comportamento uniforme no PARBASE em relação à variações no tamanho das relações envolvidas. Este trabalho também mostra o ganho no desempenho quase sempre linear em relação ao acréscimo de processadores à configuração do NCP I.

1 INTRODUÇÃO

O desempenho de aplicações de banco de dados em máquinas sequenciais vem se tornando um fator crítico para aplicações que envolvem consultas complexas e relações com grande volume de dados. O processamento de operações de banco de dados oferece várias oportunidades de paralelismo que apontam um ganho no tempo de resposta dos sistemas de banco de dados (SBD). A habilidade de se explorar um grande número de processadores conectados em paralelo é hoje uma área de pesquisa intensa. A contribuição do processamento distribuído para banco de dados está no acesso paralelo ao armazenamento das bases de dados e no processamento paralelo desses dados. A dificuldade de se sintonizar um sistema de banco de dados paralelo (SBDP), está intimamente ligada ao balanceamento de carga entre os processadores e à minimização do custo de coordenação e sincronização [Laks 90]. O panorama atual dos SBDPs tem muito a contribuir com experências práticas ou analíticas para as diversas fontes de paralelismo na gerência de bases de dados.

No sentido de se ganhar experiência na distribuição e sincronização de operações da álgebra relacional, foi implementado na máquina paralela desenvolvida na COPPE, o NCP I [Amor91], um protótipo de SBDP, o PARBASE, utilizando o ambiente de programação paralela STRAND [Fost90]. Realizaram-se várias medidas de desempenho no PARBASE onde verificou-se o ganho no desempenho sempre que o número de processadores envolvidos no processamento da consulta aumentava, além de ser observada a estabilidade do sistema em relação à variação do número de tuplas nas relações envolvidas.

Neste trabalho, são apresentadas na seção 2, as principais características e técnicas envolvidas no projeto de SBDPs atuais. A seção 3 apresenta o ambiente de desnvolvimento do protótipo PARBASE. As características, os componentes e a implementação do PARBASE são descritos na seção 4. Comentários sobre o desempenho e comportamento do PARBASE são realizados na seção 5 com o auxílio de gráficos de

tempo de resposta e de ganho em desempenho. Finalmente a seção 6 conclui o trabalho com a experiência obtida.

2 ESTAGIO ATUAL DOS SBDs PARALELOS

As pesquisas na área de paralelismo em sistemas de banco de dados surgem a partir da tecnologia de sistemas de banco de dados distribuídos e das máquinas de banco de dados. Embora essas duas áreas de conhecimento em banco de dados venham sendo exploradas há muito tempo, recentemente elas ganharam um novo impulso com a disponobilidade da tecnologia atual das arquiteturas paralelas, gerando os chamados sistemas de banco de dados paralelos (SBDP). Uma taxonomia para diversos protótipos/produtos encontrados na literatura de acordo com as diversas fontes de paralelismo mencionadas neste trabalho, pode ser encontrada em [Matt91]. Esta taxonomia visa identificar aspectos que são úteis para a comparação, descrição e desenvolvimento de SBDPs.

2.1 Categorias de SBDPs

Os SBDPs atuais caracterizam-se por seus estágios de desenvolvimento como protótipo ou produto e ainda o seu acoplamento ao hardware. Existem basicamente três categorias de concepção de SBDPs descritas a seguir:

i) máquina de bd

As máquinas paralelas de banco de dados são totalmente acopladas ao hardware, ou seja, é projetado um hardware com capacidade de processamento paralelo, específico para suportar as operações de um sistema de banco de dados. Os produtos NonStopSQL [Tand88] e DBC/12 [Alma89] são exemplos representativos dessa categoria. Como protótipos, podem ser citados os sistemas Bubba [Bora90] e Gamma [DeWi90] entre outros.

ii) servidor de bd

Os servidores de sistemas de banco de dados paralelos caracterizam-se por seu desacoplamento ao hardware. Em geral, esses servidores não utilizam um hardware específico para banco de dados, mas possuem um sistema operacional que é orientado para banco de dados. Muitas vezes, esse servidor está ligado a várias estações de trabalho que interagem com os usuários realizando um processamento local. Um exemplo típico desta categoria é o projeto do European Data Server (EDS) [Vald90].

iii) SBDP geral

Estes sistemas paralelos, são gerenciadores de banco de dados desenvolvidos para um supercomputador de uso geral, ou seja, embora o SBDP rode sobre a arquitetura de uma determinada máquina paralela, esta máquina é de uso geral e não específico para banco de dados. Como exemplo desta categoria está o protótipo XPRS de Stonebraker [Ston88].

2.2 Níveis de granularidade

Os SGBDs em geral, mais especificamente o relacional, oferecem várias oportunidades de paralelismo tanto na entrada/saída de muitos dados quanto no processamento de consultas. As pesquisas de SGBDs paralelos se concentram nas áreas de particionamento de dados - para aumentar o desempenho da entrada/saída - e no processamento de consultas. Este por sua vez pode ser dividido nos níveis de granularidade a seguir:

i) Entre consultas:

Caracteriza-se pela execução de consultas de diversos usuários concorrentemente. O objetivo da implementação paralela é de atender o maior número possível de consultas por segundo de forma concorrente.

ii) Dentro de consultas:

O objetivo é de atender o mais rápido possível ao processamento de consultas complexas e ad-hoc. Em geral, aplicações não convencionais (do tipo CAD, IA, Engenharia) caracterizam-se por possuirem poucas, longas e complexas transações exigindo assim um alto grau de paralelismo dentro da máquina para a própria transação ou consulta complexa. O paralelismo dentro da consulta está ligado à execução de operações de uma mesma consulta em paralelo.

iii) Dentro de operações:

Quando uma consulta é mapeada para as operações correspondentes que a resolvem, existem uma série de algoritmos incorporados ao SGBD que executam essas operações. Nesse nível de granularidade, procura-se paralelizar o algoritmo responsável pela resolução da operação, explorando ao máximo a disposição (particionamento) dos dados. No caso do modelo relacional, essas operções compõem a álgebra relacional, e o paralelismo dentro das operações da álgebra consiste, em geral na replicação da operação entre os processadores, e na excução da mesma operação para as partes da relação. Encontra-se na literatura um vasto material contendo algoritmos paralelos para os diversos operadores da álgebra relacional [Bitt83], [Vald84], alguns visando explorar ao máximo determinadas classes de arquiteturas como em [Rich87] e [Frie90], ou ainda explorando modelos de memória onde novos algoritmos são propostos [Chei88], [Schm89], [Murp89].

2.3 Modelos de memória

A organização da memória (principal e disco) têm influência decisiva no projeto do servidor de dados de um SBDP. Os algoritmos utilizados e principalmente a comunicação para entrada/saída costumam ser direcionados para o modelo de memória compartilhada ou para o modelo de memória distribuída (troca de mensagens). Stonebraker [Ston86] batizou esses dois modelos respectivamente de 'shared everything' (armazenamento compartilhado), já que tanto a memória principal quanto o armazenamento em disco são compartilhados por todos os processadores da arquitetura paralela e de 'shared nothing' (armazenamento não compartilhado), já que cada processador possui a sua memória local exclusiva e suas unidades de disco

também de acesso exclusivo. Existe ainda a denominação 'Shared Disk' [Bhid88b] para um modelo intermediário onde os processadores possuem memória local mas o espaço em disco é compartilhado.

Não existe um consenso em relação ao modelo mais eficiente. Alguns trabalhos analíticos [Bhid88a,b] e [Laks90] apontam a arquitetura com memória compartilhada como a solução que atinge o maior desempenho. Entretanto, vários projetos como Gamma [DeWi90], Bubba [Bora90], EDS[Vald90], entre outros apostam na expansibilidade da memória distribuída.

A seguir serão apresentadas algumas características dos modelos [Ozsu91], [Alma89], [Ston88], comparando-os do ponto de vista do projetista de um SGBD (já que para o usuário final o modelo da memória é transparente).

i) Arquitetura com memória compartilhada (Shared Everything)

A memória compartilhada é um ponto de estrangulamento em potencial. Se ocorre uma falha no esquema de interconexão da memória com os processadores, o sistema entra em pane. A arquitetura permite poucas extensões para o hardware possuindo um número limitado de processadores, uma vez que são dependentes da largura da banda ('bandwidth') da tecnologia do barramento ('bus') e da memória. Entretanto, na memória compartilhada, o particionamento dos dados é muito dinâmico pois os dados não estão distribuídos fisicamente permitindo um bom balanceamento de carga, já que num SGBD este fator está intimamente ligado à distribuição dos dados entre os processadores.

ii) Arquitetura com memória distribuída (Shared Nothing)

Permite expansibilidade para a arquitetura, o software e o hardware são projetados prevendo a extensão do número de processadores. Aumenta a disponibilidade dos dados através do isolamento de falhas e replicação. A programação do projetista do SGBD, de um modo geral, é mais complexa pois existe a necessidade de controle dos dados replicados, e obriga um controle adicional de concorrência para a cooperação de processos. Pode aumentar o desempenho aparente da entrada/saída dos dados através da realização de acessos paralelos às unidades de disco evitando o uso dos controladores de disco do processador dedicado à ES. Entretanto, o grau de paralelismo vai depender do particionamento dos dados entre os discos associados aos processadores.

2.4 Particionamento dos dados

O desempenho de uma operação está diretamente ligado com a fragmentação no armazenamento dos dados de uma base. Os algoritmos levam em consideração essa distribução no sentido de se obter o melhor balanceamento de carga entre os nós. O balanceamento é ainda mais crítico nas arquiteturas do tipo 'shared nothing' uma vez que a alocação dos dados é em geral física, realizada no momento do carregamento da base. Neste caso, para grandes volumes de dados o ideal é se executar a operação onde o dado estiver, para evitar grande tráfego na troca de mensagens. Existem dois grandes tipos de particionamento de dados o agrupado ('clustering') e o desagrupado ('declustering') [Ozsu91]. No particionamento **agrupado**, cada relação fica totalmente "contida" em um mesmo nó. Essa técnica poderá minimizar o tempo total de execução se todas as relações envolvidas estiverem no mesmo nó. Entretanto, o particionamento **desagrupado** é o mais utilizado, onde uma relação é fragmentada horizontalmente através dos nós. Neste caso, o tempo de resposta

provavelmente será minimizado, porém o tempo total poderá aumentar devido ao custo de manter a relação fragmentada através de uma determinada função.

A fragmentação horizontal pode ser feita de diversas formas e a informação sobre o particionamento da relação é armazenado no catálogo. No caso de uma arquitetura com memória compartilhada ('shared everything') os diversos fragmentos são espalhados em arquivos separados através das unidades de disco.

Na arquitetura com memória distribuída ('shared nothing') os fragmentos são distribuídos entre as unidades de disco associadas aos processadores, ou seja, a técnica de particionamento leva em conta o número de processadores com unidades de disco associadas.

As principais técnicas de fragmentação horizontal são:

i) Circular

Cada tupla é associada sequencialmente a um fragmento e os fragmentos são percorridos circularmente ('round-robin').

ii) Faixa de valores

As tuplas são fragmentadas de acordo com o valor de um determinado atributo dentro de uma faixa de valores. Por exemplo, empregados com salário entre 1.000 e 5.000 e empregados com salário acima de 5.000.

iii) Hashing

As tuplas são fragmentadas de acordo com o valor de um determinado atributo aplicado numa função de hashing. Nesta estratégia uma função randômica é utilizada para relacionar a unidade de armazenamento (o fragmento) com a tupla da relação. Por exemplo, seja uma arquitetura do tipo 'shared nothing' com 4 processadores, cada um com suas unidades de disco. Uma estratégia seria utilizar uma função randômica que gerasse valores entre 1 e 4 e as tuplas seriam então fragmentadas nesses 4 grupos ('buckets') associados aos quatro processadores.

Além das técnicas de fragmentação da relação, outras decisões tem que ser tomadas ao longo do particionamento dos dados. Essas decisões envolvem a associação da relação aos processadores [Cope88], a utilização de memória cache e a utilização de índices.

3 O AMBIENTE DE DESENVOLVIMENTO DO PARBASE

O ambiente de desenvolvimento do PARBASE pode ser dividido em duas partes. A primeira é o ambiente computacional caracterizado pela máquina paralela da COPPE/UFRJ, o NCP I. A segunda consiste no ambiente de programação englobando o sistema operacional paralelo Helios [Heli88] e o ambiente de programação paralela STRAND [Fost90], [Arti90]. O sistema operacional Helios é uma versão paralela do Unix. A vantagem da utilização do Helios está na sua portabilidade, pois sendo genérico e

rodando em diversas máquinas, facilita a integração do SBDP com outros sistemas. STRAND é uma linguagem de programação com características avançadas tanto para a programação sequencial convencional quanto para a programação paralela.

3.1 Ambiente computacional

O computador paralelo NCP I [Amor91] é um protótipo pré-industrial projetado e implementado na COPPE/UFRJ. O NCP I é baseado numa arquitetura de memória distribuída com topologia básica hipercúbica. A comunicação entre os nós de processamento é feita através dos elos seriais de comunicação à uma taxa de 20 Mbits/s. A arquitetura de um nó é mostrada na Figura 1. Como pode ser visto, cada nó possui um transputer T800 (32 bits, 25 MHz com 4 Mbytes de memória e um microprocessador i860 (64 bits, 40 MHz) com 8 Mbytes de memória, esta sendo compartilhada com o transputer. Além disso, existem 16 Kbytes de memória vetorial para troca rápida de informação entre os dois processadores. A comunicação externa é feita através de um crossbar switch e de um barramento VME. O primeiro permite a interligação do nó com outros 2048 nós, utilizando comunicação serial. Este crossbar switch permite desacoplar a arquitetura da topologia hipercúbica quando necessário. A interface VME fornece uma comunicação paralela à base de dados podendo ser utilizada também para comunicação entre nós vizinhos. Neste barramento pode ser colocado, além de controladores de disco, interface para rede e interface para outros computadores hospedeiros. A capacidade do protótipo é de 1,28 GFlops, 800 Mips e 192 Mbytes de memória.

O modelo T8 do NCP I utilizado neste trabalho, corresponde a 8 nós interligados nessa topologia hipercúbica. Cada nó contém um transputer e 2 Mbytes de memória. O computador hospedeiro é um IBM PC-AT. A capacidade do T8 é de 16 MFlops, 80 Mips e 16 Mbytes de memória.

3.2 Ambiente de programação STRAND

i)STRAND como linguagem de programação convencional

STRAND é uma linguagem declarativa e que impõe recursividade em sua programação. Possui tipos de dados estruturados como listas, tuplas e conjuntos, o que facilitou o desenvolvimento do PARBASE. Entretanto, todas as suas variáveis são univaloradas, o que acrescenta alguma dificuldade de programação. STRAND é interpretada e foi projetada para a programação multi-linguagens, ou seja, possui mecanismos de integração com Fortran e C para o desenvolvimento de código local sequencial e reaproveitamento de código. A linguagem tem sabores de Prolog e Smalltalk, mas não chega a implementar os paradigmas da programação em lógica ou da orientação a objetos.

ii)STRAND como linguagem de programação paralela

Um programa STRAND caracteriza-se por um pool de processos se comunicando através de variáveis que são argumentos dos processos. Os processos são obtidos a partir da execução de procedimentos que são escritos como regras da forma:

$r(p_1,..,p_n)$:- <guarda de cláusulas> | <corpo>.

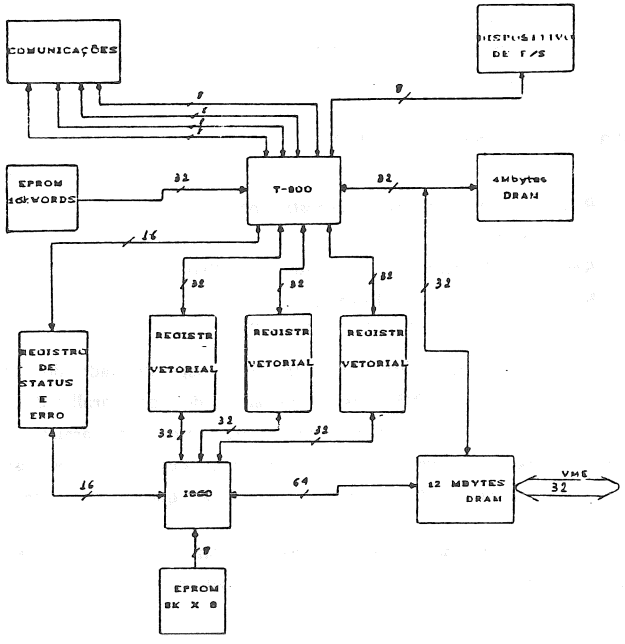

Figura 1 - Arquitetura do nó do NCP I

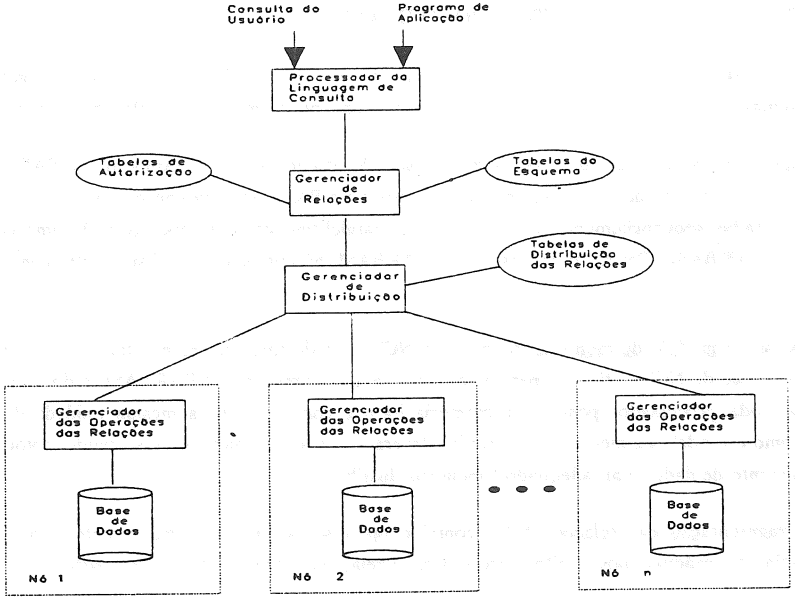

Figura 2 - Componentes do PARBASE

Onde r identifica uma regra de n parâmetros, ":-" é o símbolo de implica em, <guarda de cláusulas> são as cláusulas a serem verificadas para a execução do corpo, "|" delimita as cláusulas do corpo e o <corpo> é um conjunto de processos.

Um procedimento teste pode ser definido por:

```
teste(Numero, Resposta):- Numero >0 | Resposta:=positivo.
teste(0, Resposta):- Resposta:=zero.
teste(Numero, Resposta):- Numero <0 | Resposta:=negativo.
teste(Numero, Resposta):- otherwise | Resposta:=erro.
```

STRAND é altamente concorrente, aproveitando a disponibilidade dos processadores para executar (reduzir) os processos do pool. Não existem comandos do tipo 'parallel do', 'fork' ou 'join'. A princípio todos os processos podem ser executados concorrentemente e em paralelo, o que implica num sincronismo que é controlado pelas variáveis (os argumentos dos processos). Um processo que depende da instanciação de uma determinada variável, fica aguardando no pool até que ela receba um valor.

Outra característica interessante é a sua associação direta entre código e processadores. No caso de um SBDP, o código da seleção pode ser associado a n processadores diretamente para que a seleção seja executada em paralelo.

4 CARACTERÍSTICAS DO PROTÓTIPO PARBASE

PARBASE é um protótipo de sistema de gerência de bases de dados paralelo de uso geral que implementa alguns operadores da álgebra relacional e é independente da máquina paralela hospedeira.

O nível de granularidade do processamento paralelo de consultas mais explorado no PARBASE é o paralelismo dentro de operações. Entretanto, uma consulta no PARBASE não necessariamente terá suas operações executadas sequencialmente. É possível haver paralelismo entre as operações de uma consulta controlado pelo PARBASE. Por ser mono-usuário, o PARBASE não provê o paralelismo entre consultas de vários usuários.

Apesar do modelo de memória da máquina NCP I ser do tipo 'shared nothing' (armazenamento distribuído), a versão do NCP I utilizada para o desenvolvimento e execução do PARBASE é do tipo 'shared disk', ou seja, cada processador possui sua memória local e compartilham a mesma unidade de disco. Entretanto, como o modelo da memória da máquina deverá ser de armazenamento distribuído, optou-se por um particionamento de dados mais adequado à memória distribuída.

A fragmentação das relações é horizontal e optou-se por um desagrupamento total para as relações, ou seja, se a máquina possui oito processadores a relação é fragmentada em oito partes.

A política de distribuição de tuplas de uma relação entre os diversos fragmentos é a de faixa de valores. No momento da criação de uma relação o usuário deve especificar o atributo de distribuição e as

respectivas faixas de valores. Esta estratégia de distribuição assim como o hashing, tiram proveito da localidade conhecida para as faixas de valores e com isso atingem um ganho em desempenho.

4.1 A implementação do PARBASE

Os componentes do PARBASE estão representados na Figura 2. Essa implementação aproveita várias sugestões fornecidas por Steer [Stee89] para o desenvolvimento de um SGBD concorrente.

O usuário interage com o PARBASE através de comandos associados aos operadores da álgebra relacional, com a seguinte sintaxe:

```
select(Id-us, Atributos, Relação, Condição, Resultado)
join(Id-us, Relação1, Relação2, Condição, Resultado)
insert(Id-us, Relação, Tupla, Resultado)
delete(Id-us, Relação, Condição, Resultado)
update(Id-us, Relação, Condição, Atributos, Resultado)
create(Id-us, Relação, Chave, Especificação, Resultado)
drop(Id-us, Relação, Resultado)
```

No sentido de se facilitar o desenvolvimento do protótipo, não foi implementado um processador para analisar a linguagem de consulta. Assim a sintaxe elaborada simplifica as verificações léxicas e sintáticas no PARBASE. O usuário pode submeter um ou mais comandos através de uma lista ao PARBASE. O sistema se encarregará de prover o máximo de paralelismo entre os comandos da lista do usuário.

O Gerenciador de Relações, é responável por algumas verificações semânticas do tipo existência da relação envolvida, nível de autorização do usuário para determinadas operações, etc.

O Gerenciador de Distribuição é o responsável pela análise dos atributos envolvidos na operação e pela consequente decisão sobre quais processadores estarão envolvidos na execução da operação. O Gerenciador de Distribuição consulta as tabelas de distribuição e eventualmente impõe alguma sincronização na execução das sub-operações. Na identificação dos processadores e fragmentos para operações que envolvem condições, o Gerenciador de Distribuição verifica se somente um processador será ativado ou se todos executarão a operação, cada um com seu fragmento da relação. Ao término da operação, esse mesmo Gerente se encarrega de fazer a união da relação resultado. Quando somente um processador é envolvido na operação, os outros processadores executam as operações restantes da consulta do usuário.

No caso do operador de junção, o Gerente de Distribuição identifica se a junção pode ser executada totalmente distribuída ou se a junção será executada parcialmente distribuída. A execução da junção totalmente distribuída ocorre quando as duas relações envolvidas estão fragmentadas sobre seus respectivos atributos da condição de junção. Neste caso, as operações de junção são executadas em paralelo cada uma em um processador e é realizada a união dos resultados parciais no final. A execução da junção parcialmente distribuída ocorre em duas fases. Inicialmente o Gerenciador de Distribuição escolhe a relação de menor cardinalidade, junta todos os seus fragmentos para que em seguida a envie completa para todos os

processadores para que cada um execute a junção com a parcela da outra relação operando, associada ao seu nó.

5 ANÁLISE DO DESEMPENHO DO PARBASE

São apresentados aqui os resultados iniciais de uma avaliação do desempenho do protótipo do SBDP PARBASE na máquina paralela NCP I. Foram realizados uma série de testes com os operadores de seleção e junção onde foram analisados os tempos de resposta medidos para variações no número de tuplas das relações e no número de processadores utilizados na configuração da arquitetura.

As relações utilizadas para o benchmark são instâncias de uma base de dados real. Cada relação possui quatro atributos inteiros de 4-bytes e quatro atributos do tipo cadeia de caracteres totalizando 97 caracteres. Foram realizados testes sobre relações de 100, 1000 e 3000 tuplas. Cada relação possui dois atributos sem duplicatas e não estão relacionados entre si. Um desses atributos de valor único, foi utilizado para a chave de distribuição em todos os casos.

Todos os testes foram repetidos para configurações de 1, 4 e 8 processadores no NCP I. Os tempos foram medidos a partir do momento que o usuário submeteu a consulta ao PARBASE até a finalização da consulta. Foram incluídas neste tempo todas as passagens da consulta pelos diversos gerentes do PARBASE.

5.1 Seleção

O desempenho do PARBASE foi explorado para vários tipos de seleção sobre relações cujo tamanho também variava, baseado no benchmark do SBDP Gamma [DeWi88]. O objetivo dessas medidas foi o de analisar o comportamento do PARBASE à medida que o tamanho das relações aumentava. Em condições ideais, espera-se que o tempo de resposta aumente de acordo com uma função linear sobre o tamanho das relações operando, dado que as configurações da máquina são constantes.

Foram realizados testes sobre quatro tipos de consulta envolvendo a seleção. Em tres consultas variou-se o fator de seletividade com 0%, 1% e 10% para as relações resultado. Na quarta consulta o resultado da seleção envolvia somente uma tupla.

As tabelas 1 e 2 mostram os resultados obtidos para os diversos tipos de seleção sobre relações de tamanho 100, 1000 e 3000 tuplas para as configurações de 4 e 8 processadores ligados em hipercubo. Como as relações eram de uma base real foram realizadas pequenas adaptações para se atingir exatamente os fatores de seletividade.

	Número de tuplas da relação fonte		
Consulta	100	1000	3000
seletividade 0%	5,40	50,26	175,87
seletividade 1%	5,42	50,94	174,59
seletividade 10%	5,44	47,85	173,15
seleção 1 tupla	4,74	44,14	128,76

Tabela 1 - Consultas de seleção para 4 processadores
(todos os tempos em segundos)

	Número de tuplas da relação fonte		
Consulta	100	1000	3000
seletividade 0%	4,45	24,26	76,23
seletividade 1%	4,40	23,27	74,25
seletividade 10%	4,45	23,29	76,23
seleção 1 tupla	3,34	18,69	61,29

Tabela 2 - Consultas de seleção para 8 processadores
(todos os tempos em segundos)

Algumas conclusões podem ser obtidas das tabelas 1 e 2. Em primeiro lugar, conforme esperado, observa-se que o tempo de execução de cada consulta aumenta quase que proporcionalmente com o tamanho das relações operando para cada número de processadores testados, ou seja, 1, 4 e 8.

A única excessão ocorre para as seleções da relação de 100 tuplas quando aumentam para 1000 tuplas na configuração de 8 processadores. Nas configurações de 1 e 4, o tempo de resposta cresce em função do aumento de tuplas nas relações. Essa excessão é explicável pela má distribuição de carga que ocorreu entre a relação de cardinalidade 100 e a configuração de 8 processadores. Foi observado que não existe muito ganho em desempenho ao se utilizar muitos processadores para uma relação com poucas tuplas. Em casos assim, o preço que se paga para manter os diversos fragmentos e a comunicação entre os processadores não se traduz em custo/benefício. Esta é uma situação em que o desagrupamento total não funciona bem.

Como segunda conclusão pode ser observado o ganho em desempenho sempre que se aumenta o número de processadores e consequentemente o número de fragmentos que compõem a relação.

Cabe ainda observar que os melhores tempos de resposta estão sempre para a seleção de uma tupla, já que esta é a única consulta em que o Gerente de Distribuição do PARBASE tira proveito da distribuição da relação e não necessita percorrer toda a relação, somente o fragmento envolvido.

A Figura 3 apresenta o tempo médio de resposta para consultas com fator de seletividade 0%, 1%, 10% e seleção de 1 tupla, para a relação de 1000 tuplas em função do número de processadores utilizados. A Figura 4 apresenta a curva de ganho relativo ao tempo de processamento correspondente à Figura 3. Pode ser observado que o ganho é quase linear para os fatores de seletividade.

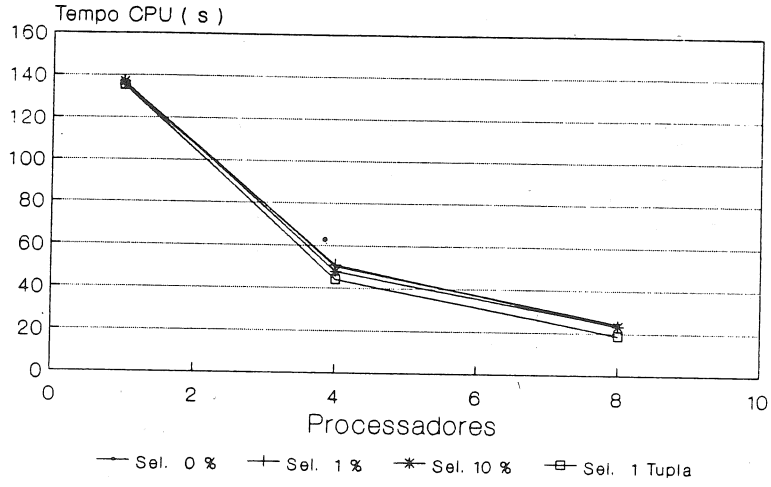

Figura 3 - Tempos de resposta base de 1000 tuplas.

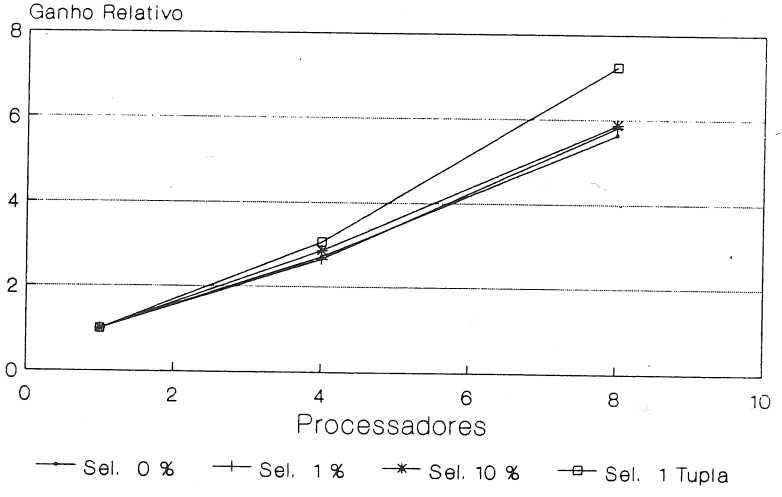

Figura 4 - Ganhos para a base de 1000 tuplas

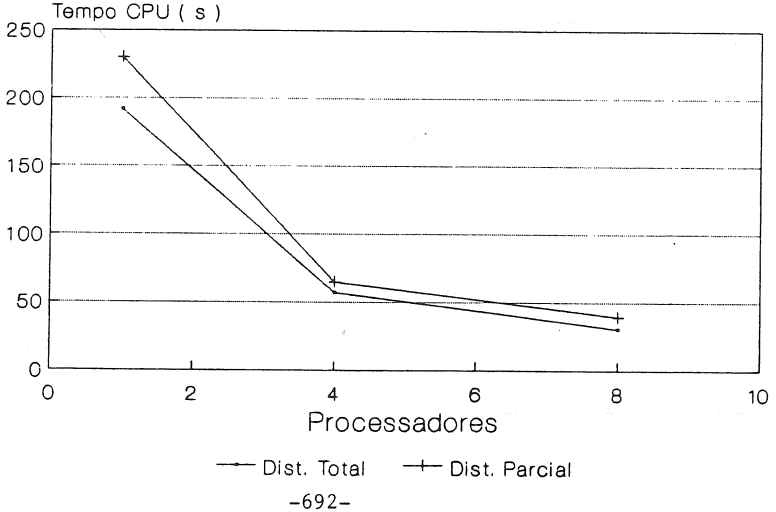

Figura 5 - Tempos de resposta junção de 100x1000 tuplas

5.2 Junção

Foram realizados diversos testes sobre o operador de junção do PARBASE e são apresentados aqui os resultados para junções ocorrendo entre tabelas de 100x1000 tuplas. O objetivo das medidas das consultas envolvendo o operador de junção foi o de analisar o desempenho do PARBASE à proporção que o número de processadores e fragmentos da relação aumentava.

Dois tipos de junção foram testados. No primeiro tipo, a junção opera sobre os atributos de distribuição das duas relações operando. Neste caso o PARBASE executa a operação de junção totalmente distribuída. No segundo tipo de teste, a junção opera sobre atributos não chave de distribuição das relações operando. Neste caso, o PARBASE executa a operação de junção em duas fases, sendo que a primeira é para a redistribuição da relação de menor cardinalidade.

Conforme era esperado, pode ser observado a melhora do desempenho através das curvas de tempo de resposta (Figura 5) e do ganho relativo. Estas curvas também mostram um tempo de resposta mais rápido para a junção totalmente distribuída. Este ganho só não é mais significativo devido ao fato da relação de menor cardinalidade conter apenas 100 tuplas.

5.3 Considerações sobre a implementação

O protótipo PARBASE foi implementado no sentido de se explorar técnicas de paralelismo para banco de dados e analisar o comportamento de um SBDP no computador paralelo NCP I. Para atingir esses objetivos foi realizada uma implementação do tipo prototipagem rápida (usufruindo de STRAND). Nessa implementação não houve preocupação com o desempenho em comparação com outros sistemas. Buscou-se a elaboração de um sistema que provê a funcionalidade mínima de um SGBD e que serve como uma plataforma onde diversas técnicas de paralelismo podem ser experimentadas, contribuindo para a construção de um SBDP que atenda necessidades reais.

Embora a implementação tenha atendido às expectativas de aceleração e estabilidade no seu comportamento, o PARBASE apresenta alguns pontos de estrangulamento, já conhecidos na fase de projeto e evidenciados nos testes realizados. Dentre esses pontos críticos, podemos citar:

AMBIENTE DE PROGRAMAÇÃO - Se por um lado a linguagem STRAND apresenta características que favorecem a prototipagem rápida, pois evita a programação de rotinas de comunicação, por outro lado, a comunicação entre os processadores pode não estar sendo realizada da maneira mais eficiente possível. Além disso a linguagem é interpretada e não foi explorada a sua facilidade de comunicação com outras linguagens. Na próxima versão do PARBASE pretende-se escrever todas as rotinas do Gerenciador de Operações sobre as relações em C, o que deverá diminuir o tempo de acesso aos dados.

ARMAZENAMENTO SECUNDÁRIO - O fato do NCP I estar utilizando na atual versão o disco rígido de um PC para o armazenamento secundário, faz com que a comunicação da entrada e saída seja um ponto de estrangulamento, pois todos os processadores competem para obter o mesmo recurso. Somado a isso está a ausência de métodos de acesso próprios ao PARBASE e de um esquema de bufferização.

ÍNDICES - A ausência de índices no processamento de consultas é um fator limitante no desempenho do PARBASE para diversas consultas. Segundo DeWitt [DeWi 88] a utilização de índices no SBDP Gamma fez com que o tempo de resposta fosse reduzido em aproximadamente 50% sendo observado em alguns casos reduções de até 25%. Cabe salientar que índices locais podem ser facilmente implementados no PARBASE.

ANALISADOR DA LINGUAGEM DE CONSULTA - Optou-se por uma forma de interação com o PARBASE intermediária entre uma linguagem de consulta tipo SQL e um código de máquina interpretável pelo Gerente de Relações. Assim, evitou-se por um lado o desenvolvimento de um analisador léxico e sintático, mas por outro lado algumas análises e transformações semânticas do código foram repetidas ao longo de diversas fase do PARBASE.

6 CONCLUSõES

Foram levantadas na seção 2 diversas fontes de paralelismo na gerência de bases de dados. A proposta do PARBASE difere basicamente das outras por dois fatores. O primeiro é a sua independência da máquina paralela hospedeira e do sistema operacional. O segundo fator é a obtenção das medidas de desempenho na plataforma real onde se deseja desenvolver o SBDP produto.

Optou-se pelo desenvolvimento de um protótipo simples com as características básicas de paralelismo, sobre o qual várias técnicas alternativas podem ser avaliadas e exploradas. Os gerenciadores do PARBASE dão tratamento uniforme para todas as operações implementadas facilitando a modificação e extensão do sistema. Várias alterações foram propostas no item 5.3 no sentido de contribuir para o desempenho do PARBASE em sua próxima versão.

Foram obtidas diversas medidas de desempenho e reportadas aqui algumas aferições para as operações de seleção e junção, já que representam respectivamente operações totalmente distribuídas e diretamente paralelizáveis e operações parcialmente distribuídas. Os resultados foram de acordo com o esperado, ou seja, verificou-se um crescimento uniforme no sistema a partir do tamanho das relações e um ganho no desempenho quase sempre linear em relação ao acréscimo dos processadores nos casos analisados.

Finalmente, o fato de se possuir um protótipo que não foi desenvolvido especificamente para um determinado sistema operacional ou máquina paralela, permite que: i) novas medidas de desempenho possam ser realizadas e comparadas em outras máquinas, ii) haja mais independência dos dados, e iii) a construção e migração de aplicações para o SBDP sejam facilitadas.

7 REFERÊNCIAS

[Alma89] Almasi,G.S., Gottlieb,A. "Highly Parallel Computing" Benjamin Cummings, 1989.

[Amor91] Amorim,C.L. e equipe "O computador paralelo NCP I" Relatório interno, COPPE-Sistemas/UFRJ, março 1991.

[Arti90] Artificial Intelligence Limited "STRAND^{88} User Manual" Buckingham Release, June 1990.

[Bhid88a] Bhide,A., Stonebraker, M. "A Performance Comparison of Two Architectures For Fast Transaction Processing" Proceedings of the 4th International Conference on Data Engineering, IEEE, Los Angeles, Feb 1988.

[Bhid88b] Bhide,A. "An Analysis of Three Transaction Processing Architectures" Proceedings of the 14th International Conference on Very Large Data Bases, pp. 339-350, Los Angeles, 1988.

[Bitt83] Bitton,D. Boral,H. DeWitt,D. Wilkinson,K. "Parallel Algorithms for the Execution of Relational Database Operations" ACM Transactions on Database Systems, v.8(3), pp. 325-353, Sept 1983.

[Bora90] Boral,H. et al. "Prototyping Bubba, A Highly Parallel Database System" IEEE Transactions on Knowledge an Data Engineering, v.2(1), pp. 4-24, March 1990.

[Chei88] Cheiney,J-P. Maindreville,C. "A Parallel Transitive Closure Algorithm Using a Hash-Based Clustering" INRIA Research Repport 950, Dec 1988.

[Cope88] Copeland,G. Alexander,W., Boughter,E., Keller,T. "Data Placemente in Bubba" Proceedings ACM SIGMOD International Conference on Management of Data, pp.99-108, Chicago, June 1988.

[DeWi88] DeWitt,D.J. Ghandeharizadeh,S. Schneider,D. "A Performance Analysis od the Gamma Database Machine" Proceedings ACM SIGMOD International Conference on Management of Data, pp.350-360, Chicago, June 1988.

[DeWi90] DeWitt,D. et al. "The GAMMA Database Machine Project" IEEE Transactions on Knowledge an Data Engineering, v.2(1), pp. 44-62, March 1990.

[Heli88] Helios Group at Perihelion Software Limited "The Helios Operating System" Distributed Software Ltd, Version 1, Release 1.0, August 1988.

[Fost90] Foster I., Taylor S. "Strand: New Concepts in Parallel Programming", Prentice Hall, 1990.

[Frie90] Frieder,O. "Multiprocessor Algorithms for Relational-Database Operators on Hypercube Systems" IEEE Computer, v.23(11), pp. 13-29, Nov 1990.

[Laks90] Lakshmi,M.S. Yu,P.S. "Effectiveness of parallel processing in database systems", Computer Systems Science and Engineering, v.5(2) pp.73-80, April 1990.

[Matt91] Mattoso, M.L.Q."Paralelismo e Banco de Dados: Um Levantamento" Relatório Técnico da COPPE/UFRJ, a ser publicado.

[Murp89] Murphy, M.C. Rotem,D. "Effective Resource Utilization for Multiprocessor Join Execution" Proceedings of the 15th International Conference on Very Large Data Bases, pp. 67-75, Amsterdam, August 1989.

[Rich87] Richardson,P.J. Lu,H. Mikkilineni,K. "Design and Evaluation od Parallel Pipelined Join Algorithms" Proceedings ACM SIGMOD International Conference on Management of Data, pp.399-409, San Francisco, May 1987.

[Schn89] Schneider,D. DeWitt,D.J. "A Performance Analysis of Four Parallel Join Algorithms in a Shared-Nothing Multiprocessor Environment" Proceedings ACM SIGMOD International Conference on Management of Data, pp.110-121, Portland, June 1989.

[Stee89] SteerValduriez,P. "Distributed and Parallel Database Systems", Apresentação de Tutorial, 5. Simpósio Brasileiro de Banco de Dados, Rio de Janeiro, Abril 1990.

[Ston86] Stonebraker,M. "The Case for Shared Nothing" IEEE Database Engineering, v.9(1), March 1986.

[Ston88] Stonebraker,M. "The Design of XPRS" Proceedings of the 14th International Conference on Very Large Data Bases, pp. 339-350, Los Angeles, 1988.

[Tand88] Tandem Performance Group "A Benchmark of NonStop SQL on the Debit Credit Transaction" Proceedings ACM SIGMOD International Conference on Management of Data, pp.337-341, Chicago, June 1988.

[Vald84] Valduriez,P. Boral,H. "Join and Semijoin Algorithms for a Multiprocessor Database Machine" ACM trans. on Database Systems, v.9(1), March 1984

[Vald90] Valduriez,P. "Query Processing in the EDS Parallel Database System", Apresentação de Tutorial, 5. Simpósio Brasileiro de Banco de Dados, pp. 2-14, Rio de Janeiro, Abril 1990.

[Ozsu91] Ozsu,M. Valduriez,P. "Principles of Distributed Systems", Prentice-Hall, 1991.