Michelle Santos Sá

Uma Abordagem Online para Comparação de Junções Paralelas

Dissertação apresentada ao Departamento de Informática da PUC/RJ como parte dos requisitos para obtenção do título de Mestre em Informática: Ciência da Computação.
Orientador: Sérgio Lifschitz

Departamento de Informática

Pontifícia Universidade Católica do Rio de Janeiro

Rio de Janeiro, 15 de dezembro de 2000

Aos meus pais

Agradecimentos

À CAPES – Coordenação de Aperfeiçoamento de Pessoal de Nível Superior, pelo auxílio financeiro.

Ao professor Sérgio Lifschitz pela sua orientação, amizade e empenho na leitura e revisão do texto.

Ao professor Marcus Poggi pela colaboração no desenvolvimento da idéia proposta nesta dissertação.

À s professoras Marta Mattoso e Elvira Uchôa pela participação na banca de avaliação.

A José Antônio Macêdo e Flavio Freitas, amigos e colegas de trabalho envolvidos diretamente no desenrolar desta dissertação, sem os quais a implementação deste trabalho seria muito mais difícil. Agradecimentos especiais ao José Antônio por seus comentários sempre tão cheios de humor e por acreditar que tudo é possível.

À Melissa Lemos, amiga de todas as horas, pelo constante apoio e por seus conhecimentos sobre a ferramenta Microsoft Word.

Ao Luiz Fernando Bessa Seibel por seu constante otimismo e por sua ajuda na finalização do texto.

Aos meus pais, Orlando e Vera pelo amor a mim dedicado e por acreditarem em meus sonhos, em cuja conquista sempre me apoiaram. Ao meu sobrinho Danillo Sá pelo companheirismo e a minha prima Fabiana Sá pela sua constante amizade.

Aos inúmeros amigos, colegas, professores e funcionários que a seu modo me ajudaram direta ou indiretamente na realização deste trabalho.

A Deus, por me iluminar e guiar através de todas as oportunidades com que tenho sido agraciada.

Resumo

Diferentes abordagens para execução em paralelo do operador de junção relacional foram propostas na literatura. Para que as estratégias pudessem ser comparadas, foram desenvolvidas implementações para avaliação em ambientes paralelos variados. Entretanto, não existe um consenso em relação ao comportamento das estratégias, com ou sem balanceamento de carga, em relação às diversas condições reais onde estas estratégias são executadas. Esta dissertação tem por objetivo abordar o problema da junção paralela sob o ponto de vista da teoria de algoritmos online, utilizando a análise competitiva para avaliar e comparar o comportamento das estratégias existentes com relação às estratégias (offline) ótimas. Ao invés de um estudo analítico, como é comum no contexto online, são realizadas nesse trabalho uma implementação e uma simulação para avaliar uma estratégia online de junção paralela com balanceamento de carga quanto ao equilíbrio de carga de trabalho. Como adversários, são considerados o desvio nos valores dos atributos de junção e uma variação no número de processadores paralelos disponíveis.

Abstract

Different approaches for the parallel execution of the relational join operator were proposed in the literature. Implementations were developed in order to compare and to evaluate these strategies in a parallel environment. However, there is no agreement concerning the behavior of the strategies, with or without load balancing, with respect to several real conditions where these strategies are executed. This work presents an approach to the parallel join problem from the online algorithms theory point of view. We use competitive analysis to evaluate and compare the behavior of strategies compared to the offline (optimal) one. Instead of an analytical study, common to the online context, this dissertation presents an implementation and a simulation to evaluate an online parallel join strategy with load balancing with respect to workload distribution. As adversaries we consider a variation in the number of parallel processors available and the skew on join attribute data values.

Índice

Índice de Figuras

Capítulo 1

Introdução

A necessidade de se executar tarefas cada vez mais complexas, numerosas ou demoradas motivou a busca por sistemas compostos de múltiplos processadores. Estes sistemas deveriam ser capazes de dividir uma tarefa e executar as partes em paralelo (sempre que possível) diminuindo, assim, o tempo de execução. Estes fatos ocorrem não apenas no contexto de processamento de algoritmos como também no contexto de Sistemas Gerenciadores de Bancos de Dados (SGBDs).

Como uma das operações mais complexas dos SGBDs é execução da junção, nada mais natural do que realizar estudos sobre as formas de se executar a operação de junção em paralelo. Diversas estratégias de execução da junção em paralelo foram propostas na literatura [SD89][LY90] [DNS+92][LT94][LL98].

Para que essas estratégias pudessem ser comparadas, foram desenvolvidas diversas implementações de ambientes paralelos de avaliação. Alguns ambientes foram desenvolvidos para comparar uma determinada estratégia nova com outras [DNS+92] [LT94], enquanto outros foram desenvolvidos para comparar estratégias já existentes [SD89][HTY95]. Pelo fato de cada uma das diferentes estratégias propostas utiliza plataformas de hardware e software diferentes, torna-se difícil realizar a comparação de uma estratégia com a outra.

O trabalho apresentado em [SD89] foi o primeiro a tentar unificar em um único ambiente os algoritmos mais usados até então. Porém, ainda não existe um consenso geral em relação ao comportamento das diversas estratégias de execução de junção paralela (com ou sem balanceamento de carga) em relação às diversas condições dos ambientes reais onde estas estratégias deverão ser executadas.

Esta dissertação tem como objetivo abordar o problema da junção paralela sob o ponto de vista da teoria de algoritmos online, utilizando a análise competitiva [Alb97][AL97], técnica bastante divulgada no contexto online, para avaliar o comportamento das estratégias de execução paralela em relação à distribuição da carga de trabalho.

O artigo [LLP99] foi a principal motivação para este trabalho pois nele foram levantadas questões como, por exemplo, a utilização da análise competitiva para comparar diferentes estratégias para execução de junção paralela. Ao invés de um estudo analítico, como é comum no contexto online, este trabalho analisará experimentalmente a competitividade de uma estratégia online de junção paralela com balanceamento de carga em relação a uma outra considerada ótima (ou offline). Esta última será executada tendo um conhecimento completo sobre todas as informações relevantes para a distribuição de carga.

A estratégia de junção paralela com balanceamento de carga utilizada será uma implementação da arquitetura VAST-PJ [LL98] cujas caracteríticas principais são o tratamento preventivo do balanceamento de carga e a variação do tamanho da tarefa demandada a cada nó.

Esta dissertação está estruturada como descrito a seguir.

O Capítulo 2 apresenta o contexto de junções paralelas e desvios de dados destacando o fato de que a teoria de algoritmos online e a análise competitiva [Alb97] [AL97] podem ser utilizadas para se definir uma metodologia para avaliar o desempenho de uma determinada estratégia de execução de junção paralela.

Para apresentar uma metodologia de comparação, o Capítulo 3 caracteriza o problema da junção paralela sob o ponto de vista online. A distribuição offline (que é por construção ótima) utilizada na comparação proposta será obtida através da divisão da carga total de trabalho entre os processadores disponíveis. O capítulo propõe ainda uma lista de fatores considerados importantes para realizar a comparação entre uma estratégia online de junção paralela baseada na VAST-PJ e a estratégia offline proposta.

O Capítulo 4 relata um estudo de caso desenvolvido para ilustrar a aplicabilidade da análise competitiva no contexto de comparação de estratégias de junção paralela. Para isso, utilizou-se a métrica, definida no Capítulo 3, para analisar a distribuição da carga de trabalho total entre os processadores envolvidos na execução da junção.

Finalmente, o Capítulo 5 apresenta as conclusões, explicita as contribuições e relaciona uma lista de trabalhos futuros a serem realizados.

Capítulo 2

Contexto e Motivação

Este capítulo apresenta inicialmente o contexto de junções paralelas e desvios de dados ressaltando o fato de que, conforme foi proposto em [LLP99], a teoria de algoritmos online e a análise competitiva podem ser utilizadas para que seja definida uma metodologia para avaliar o desempenho de uma determinada estratégia de execução de junção paralela.

Devido à necessidade de se entender melhor as características dos problemas online, são definidos neste capítulo os principais conceitos da teoria de algoritmos online e da análise competitiva, tendo como exemplo o problema de paginação. Por último, são brevemente descritos alguns artigos que tratam de assuntos relacionados como algoritmos online para balanceamento de carga, algoritmos online para tratamento de desvio em junções paralelas e técnicas utilizadas na literatura para comparar estratégias de junção paralela.

2.1. Junções Paralelas e Desvios de Dados

A operação de junção é uma das mais importantes e mais caras operações realizadas pelos SGBDs. Podem ser obtidas melhorias significativas no tempo de resposta através da paralelização da junção, em um ambiente composto por **p** processadores que atuem cooperativamente e coordenadamente [WDJ91][DG92]. Em geral, o processamento da junção envolve três passos: decomposição da junção em **n** tarefas, atribuição das **n** tarefas aos **p** processadores e montagem dos resultados parciais obtidos destes processadores [LT94]. Supondo que o objetivo seja efetuar a junção entre duas relações, uma decomposição razoável desta operação em **n** tarefas corresponde a executar a junção de sub-relações das duas relações operandas, onde estas sub-relações são obtidas através de uma fragmentação horizontal das relações principais [DG92].

O tempo de resposta de um algoritmo paralelo é dado pelo tempo do processador que por último encerra sua tarefa ou grupo de tarefas, mais o tempo para construir o resultado global, se necessário. Portanto, minimizar este tempo de resposta é uma questão associada à distribuição adequada das tarefas entre os processadores, visando o balanceamento de carga e o aproveitamento máximo dos recursos computacionais

disponíveis. Entretanto, não basta apenas definir uma distribuição equilibrada de trabalho entre os processadores para que todos terminem as suas tarefas em tempos aproximadamente iguais pois, em um ambiente paralelo multi-usuário, o nível de concorrência dos diversos processadores não é necessariamente o mesmo, o que pode prejudicar o tempo total paralelo [LY90] [SD89].

Um outro problema a ser considerado é a não-uniformidade na distribuição dos valores dos atributos de junção. Quando isto ocorre, há variações no tamanho dos subconjuntos de tarefas gerados, fazendo com que os trabalhos atribuídos a cada processador não sejam equivalentes. Este tema foi inicialmente abordado em [LY90] [SD89], dando origem a vários trabalhos relacionados ao fenômeno, genericamente denominado de desvio de dados (*data skew*) ou, simplesmente, desvio.

Em [WDJ91] o conceito de desvio é formalizado, separando o desvio relacionado aos dados (*intrinsic skew*) daquele decorrente do processo de particionamento *(partitioning skew*) visando o paralelismo. Quando o trabalho para realizar a junção é particionado entre vários processadores, o desvio nos valores dos atributos de junção (*AVS – attribute value skew*) das relações operandas pode causar um desequilíbrio significativo na carga de trabalho associada a cada um dos processadores.

Sendo assim, observa-se que a solução para o problema do balanceamento de carga não se resume à distribuição equilibrada de subconjuntos de dados aos processadores pois não necessariamente isso implicaria em uma carga de trabalho equilibrada. Além disso, deve-se evitar buscar um equilíbrio perfeito de carga se o custo para garantir isso for maior do que o custo devido a um pequeno desvio nas cargas de trabalho. Ou seja, o objetivo é minimizar o tempo paralelo, não balancear carga [LPR97].

2.2. Comparação de Estratégias

Apesar das diversas estratégias de junção paralela com ou sem balanceamento de carga, não existe um consenso quanto a uma metodologia de comparação destas estratégias e, consequentemente, nem quanto a qual delas deva ser selecionada por um otimizador de consultas na prática [LLP99].

É interessante lembrar que para o caso dos ambientes não paralelos, mesmo com um número de algoritmos sequenciais de junção disponíveis (decorrentes basicamente de

variações dos três tipos básicos: iterações aninhadas; ordenação e fusão; e *hash* [ME92]), dados quaisquer dois algoritmos é possível decidir (para um determinado conjunto de dados) qual deles terá melhor desempenho, baseando-se, por exemplo, no número estimado de comparações de tuplas e em estatísticas e dados da metabase de SGBDs.

A principal motivação desta dissertação foi o artigo [LLP99], onde é apresentada a idéia de utilizar a teoria de algoritmos online [Alb97] para se estudar o problema da junção paralela com balanceamento de carga. O artigo sugere que seja utilizada a análise competitiva [AL97][Alb97] para comparar as diferentes estratégias de junção paralela existentes na literatura que possam ser consideradas online.

Esta dissertação tem por objetivo utilizar a análise competitiva para avaliar o comportamento de uma estratégia de junção paralela com balanceamento de carga – VAST-PJ [LL98] – visando ilustrar e motivar o uso da teoria de algoritmos online no contexto de comparação de estratégias paralelas de junção. Isto estaria sinalizando, com uma abordagem prática, qual estratégia de balanceamento de carga os otimizadores poderiam vir a escolher no momento da geração do plano de execução de uma consulta.

2.3. Teoria de Algoritmos Online

Esta seção tem por objetivo apresentar os principais conceitos sobre algoritmos online e análise competitiva necessários para o entendimento desta dissertação.

O projeto tradicional de algoritmos assume que uma determinada aplicação deve ter um completo conhecimento da entrada de dados antes que qualquer resultado seja fornecido. Esta suposição, entretanto, nem sempre é verdadeira nas aplicações práticas e muitos dos problemas algorítmicos têm características online, isto é, a entrada de dados está apenas parcialmente disponível, pois alguns dados relevantes de entrada não são conhecidos no momento em que deve ser gerada uma resposta. Um algoritmo online precisa determinar os passos necessários para se gerar um resultado sem conhecimento completo da entrada de dados.

Sendo assim, um algoritmo online é aquele que percebe a entrada de dados de forma incremental, ou seja, uma parte de cada vez. Em resposta a cada porção de entrada, o algoritmo deve gerar uma saída sem conhecer a próxima porção. Alguns problemas online básicos foram estudados extensivamente, como por exemplo: paginação

[FKL+91] (descrito na próxima seção); e k-servidores [MMS90], problema que consiste em planejar o deslocamento de k servidores móveis que residem em um determinado espaço S, e que devem ser enviados para um ponto específico de S quando lá surge uma determinada requisição. Além destes, muitos outros problemas online foram investigados em outras áreas de aplicação como por exemplo, escalonamento e balanceamento de carga [Sga98][Aza98], os sistemas para medida de tarefas [BLS92] e o problema de roteamento [Leo98].

Em suma, a idéia básica da teoria dos algoritmos online está relacionada com a necessidade de se tomar determinadas decisões sem que as situações futuras sejam conhecidas. De forma geral, os problemas online são aqueles que precisam gerar resultados, em tempo de execução, sem conhecer todas as informações relevantes para que este resultado seja o melhor possível [Alb97].

O problema online que será tratado nesta dissertação será a execução da junção em paralelo com balanceamento de carga. As características online deste problema serão descritas no Capítulo 3.

2.3.1. Um exemplo de Problema Online - Paginação

Para esclarecer melhor a idéia dos problemas online, considere o problema de paginação, ilustrado na Figura 1, um dos problemas online mais fundamentais. No caso, supõe-se um sistema de memória de dois níveis. No primeiro nível, uma memória menor e rápida e no segundo uma outra memória maior e lenta. Normalmente, o primeiro nível é a memória principal ou RAM – *Random Access Memory* – e o segundo nível é a memória secundária, disco rígido, CDs ou fitas. Por convenção, os dados são organizados nas memórias como um conjunto de páginas ou blocos de dados, onde cada página é um espaço físico contíguo de *bytes*. Uma página é a unidade de transferência de dados entre os dois níveis de memória.

Um algoritmo de paginação consiste na manutenção estratégica de páginas de tal maneira que elas estejam sempre que possível presentes na memória rápida quando ocorrer uma requisição de página. O principal objetivo de um algoritmo de paginação é minimizar o número de transferência de páginas entre a memória lenta e a memória rápida. A situação em que uma página requisitada já está presente na memória rápida é chamada presença de página (*page hit*). Por outro lado, quando a página requisitada não estiver na memória

rápida a situação é chamada de falta de página (*page fault*). Neste último caso, não existindo páginas livres na memória rápida, a página desejada somente poderá ser lida da memória lenta para a memória rápida depois que uma página for escolhida para sair da memória rápida.

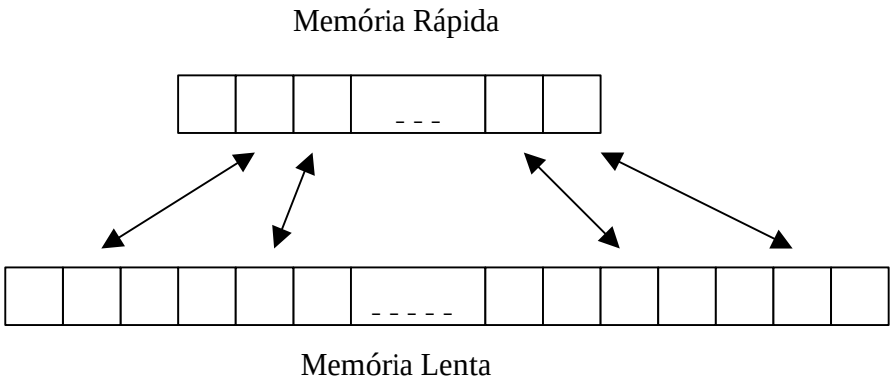

Figura 1: Representação gráfica do problema de paginação

A característica online deste problema está no fato de que o algoritmo de paginação deve manter referências ativas às páginas na memória rápida sem conhecer quais páginas serão requisitadas no futuro. Isto é, quando se decide qual página deve sair da memória rápida, não se sabe se no futuro esta página será novamente requisitada, gerando uma falta de página. Idealmente, as páginas devem ser escolhidas para sair da memória rápida quando elas não mais forem úteis para os processamentos.

Como é mostrado na Figura 1, existem dois níveis de memória, e cada nível está dividido em páginas de tamanho fixo. Algumas páginas da memória lenta (grande) estão replicadas na memória rápida (pequena) enquanto que todas as páginas estão localizadas na memória lenta (grande). Se uma página requisitada está na memória rápida então não é preciso fazer nada; caso contrário, ocorre uma falta de página e o sistema se depara com o problema de decidir qual página da memória rápida deve sair para que a página solicitada seja disponibilizada. O custo a ser minimizado é o número de falta de páginas. Um algoritmo online para este problema decide qual página será removida da memória rápida quando ocorre uma falta de página sem qualquer informação das requisições subsequentes de páginas.

A seguir são citadas algumas das possíveis estratégias online para este problema [Bar97]:

- **LIFO** (*Last-In-First-Out*): remove a última página armazenada na memória rápida.

- **FIFO** (*First-In-First-Out*): remove a que está há mais tempo na memória rápida.

- **LRU** (*Least-Recently-Used*): substitui a página acessada há mais tempo.

- **MRU** (*Most-Recently-Used*): substitui a página acessada há menos tempo.

- **LFU** (*Least-Frequently-Used*): substitui a página menos acessada.

- **FWF** (*Flush-When-Full*): nessa estratégia, marca-se uma página quando ela é acessada. Sempre que todas as páginas na memória rápida estiverem marcadas, todas são desmarcadas. Ao ocorrer uma falta de página, qualquer uma das páginas desmarcadas é removida da memória rápida e substituída pela página solicitada que, consequentemente, será marcada.

2.3.2. Formalizando Algoritmos Online

Formalmente um algoritmo online A é apresentado como uma sequência de requisições ? = ?$_{(1)}$, ?$_{(2)}$, ?$_{(3)}$, ... , ?$_{(m)}$. As requisições ?$_{(t)}$, 1 ? t ? m, precisam ser atendidas em sua ordem de ocorrência. Mais especificamente, quando o algoritmo A atende à requisição ?$_{(t)}$, ele não conhece qualquer requisição ?$_{(t´)}$ com t´ > t. O atendimento de qualquer requisição possui um custo e o objetivo é minimizar o custo total pago pela sequência inteira de requisições [AL97].

Para o problema de paginação, ? é representado pela sequência de requisições de páginas ocorridas através do tempo t. O custo é representado pelo número de falta de páginas, pois quando isto ocorre, uma página da memória rápida deve ser substituída por outra da memória lenta. O objetivo é minimizar o número de falta de páginas, minimizando desta forma o número de páginas que devem ser transferidas da memória lenta para a memória rápida.

A teoria de algoritmos online tem mostrado que a análise competitiva é uma ferramenta poderosa para avaliar o desempenho dos algoritmos online. Isto será discutido com mais detalhes a seguir.

2.3.3. Análise Competitiva

Em [AL97], a análise competitiva é definida como a comparação entre um algoritmo online A e um algoritmo offline (ótimo) OPT. Um algoritmo offline é aquele que conhece sequência completa dos dados de entrada e, desta forma, pode gerar uma saída ótima (gerar uma saída com o menor custo). O algoritmo offline é sempre considerado como sendo um algoritmo ótimo. Assim, quanto mais um algoritmo online se aproxima da solução ótima, mais competitivo este algoritmo é.

Para o caso do problema de paginação, por exemplo, o algoritmo offline é aquele que conhece antecipadamente a sequência completa de requisições de páginas que serão feitas e, assim, pode definir da melhor forma possível quais as páginas que devem ser retiradas da memória rápida para minimizar o número de falta de páginas.

Mais formalmente, dada uma sequência de entrada ?, tem-se que $C_A(?)$ e $C_{OPT}(?)$ denotam o custo associado aos algoritmos A e OPT respectivamente, para o processamento de ?. O algoritmo A é chamado **c**-competitivo se e somente se existe constantes **a** tal que

$$C_A(?) \leq c.C_{OPT}(?) + a \text{, onde } c > 1$$

para toda sequência de entrada ?. O fator **c** é chamado de razão de competitividade [Alb97].

A competitividade de um algoritmo online é definida em relação a um adversário. O adversário pode ser considerado como um outro algoritmo que constrói uma sequência ? de entrada para o algoritmo online A.

Utilizando ainda o exemplo de paginação, foi mostrado em [FKL+91][Goe94] que as estratégias online de paginação LRU e FIFO são k-competitivas, onde k é o número de páginas na memória rápida. Sendo assim, $C_{LRU}(?) \leq k.C_{OPT}(?)$, o que também é válido para a estratégia FIFO.

2.3.4. Adversários

A análise competitiva pode também ser vista como um jogo onde existe um jogador que está realmente executando o algoritmo sob análise e existe um adversário que tenta

confrontar seu oponente com condições que geram a pior saída do algoritmo. Neste caso, o adversário sempre obtém vantagem, pois ele sempre conhece qual algoritmo seu oponente está executando [LLP99].

Basicamente, a idéia por trás dos adversários dos algoritmos online é gerar condições que desafiem sua habilidade para obter um resultado eficiente. Os adversários podem ser vistos como algoritmos que criam diferentes situações e manipulam os parâmetros da plataforma – antes ou durante a execução do algoritmo.

A razão de competitividade de um algoritmo online A é definida em relação a um adversário. O adversário gera uma sequência de requisições ? e também deve atender ?. Na construção de ?, o adversário sempre conhece a descrição de A. As questões fundamentais aqui são: quando as requisições devem ser geradas e se é permitido ao adversário ver o resultado das escolhas aleatórias feitas por A sobre as requisições anteriores.

Em [Alb97] foram definidos diferentes tipo de adversários: adversário óbvio e adversário adaptativo, que é subdividido em adversário adaptativo online e adversário adaptativo offline. Estes tipos de adversários são brevemente explicados a seguir.

Adversário ó bvio:

O adversário óbvio deve gerar uma sequência completa de entrada antes que qualquer requisição seja atendida pelo algoritmo online. Consequentemente, este adversário conhece o custo do algoritmo ótimo para esta sequência.

Adversário Adaptativo Online

Este adversário deve observar o algoritmo online e gerar a próxima requisição baseada nas respostas do algoritmo para todas as requisições anteriores. Sendo assim, este tipo de adversário é obrigado a atender cada requisição online sem conhecer a escolha aleatória feita pelo algoritmo online no presente momento ou em alguma requisição futura.

Adversário Adaptativo Offline

Da mesma forma que o adversário adaptativo online, o adaptativo offline gera uma sequência de requisições adaptativamente. Entretanto, ele conhece o custo offline ótimo para esta sequência.

Para o caso de paginação, um adversário deve gerar várias sequências de requisições de páginas. O algoritmo que gera a sequência de entrada pode ou não conhecer a estratégia utilizada pelo algoritmo A.

2.4. Trabalhos Relacionados

Esta seção apresenta alguns resumos de trabalhos publicados que relacionam problemas de SGBDs com problemas online. Além disso, comenta-se sobre trabalhos de comparação de junção paralela encontrados na literatura.

2.4.1. Algoritmos Online para Tratamento de Desvio na Junção Paralela

Em [SY93], tem-se por objetivo realizar o balanceamento de carga em uma arquitetura paralela onde a memória e o disco são locais a cada um dos processadores. É importante ressaltar que o termo balanceamento de carga, no contexto do artigo, denota o tratamento do desvio na distribuição dos dados e não o balanceamento de carga entre múltiplos processadores. Isto quer dizer que o algoritmo proposto deverá tentar dividir os intervalos de valores entre os processadores existentes.

A operação de junção é vista como um conjunto de tarefas, onde cada uma deve associar tuplas das duas relações que possuam o mesmo valor no atributo de junção. Se os valores do atributo de junção estão contidos em uma faixa de um determinado domínio, diz-se que a tarefa corresponde a um intervalo de valores. Se a tarefa correspondente a um intervalo de valores do atributo de junção é muito grande, ela pode ser dividida. Porém, este particionamento da tarefa gera uma carga extra de trabalho.

O artigo considera que **p** processadores estão disponíveis para executar uma junção e que o algoritmo de balanceamento de carga deve tentar atribuir intervalos de valores a cada um destes **p** processadores a fim de que a carga de trabalho associada a cada um deles seja aproximadamente a mesma.

Define-se então W_t como sendo a carga de trabalho para realizar a junção, incluindo CPU e E/S. W_p denota o trabalho associado a cada processador para o caso onde o balanceamento é perfeito, ou seja, $W_p = W_t/p$. Denota-se por d a tolerância permitida para a diferença de carga de trabalho entre os processadores. Para o caso onde o balanceamento é perfeito, $d = 0$. Desta forma, o algoritmo de balanceamento de carga

deve atribuir as cargas de trabalho aos **p** processadores de forma que nenhum processador exceda (1 + d)W_p. Esta definição de desequilíbrio de carga de trabalho é similar à definição de desvio em [WDJ91].

O primeiro passo consiste na construção de um histograma de dados para as duas relações operandas. Estes histogramas são considerados como dados de entrada para o algoritmo online de balanceamento de carga pois eles são fundamentais para se determinar os intervalos dos atributos de junção associados a cada processador. É importante ressaltar que um intervalo pode consistir de um único valor ou de múltiplos valores distintos. Em seguida são atribuídos intervalos de valores a cada um dos processadores e as tuplas das relações operandas são enviadas pelos processadores de armazenamento aos seus respectivos processadores de junção de acordo com a distribuição de intervalos determinada pelo algoritmo.

Baseado nas definições acima, foram implementados em [SY93] dois algoritmos para balanceamento de carga na junção paralela. O primeiro é chamado algoritmo online puro (OP) e o segundo é chamado de algoritmo online/offline (OO). Estes algoritmos são descritos brevemente a seguir.

O algoritmo online puro é aquele que processa os dados do histograma à medida em que eles são gerados. Para isso, define-se que W_{seen} é o trabalho conhecido (e posteriormente atribuído) pelo algoritmo OP. Inicialmente o W_{seen} é igual a zero. Considere também que I é o trabalho do próximo intervalo a ser atribuído e que W_i denota o trabalho no processador i antes que I seja atribuído e, finalmente, que W_i' é o trabalho no mesmo processador depois que I é atribuído. Em OP, diz-se que o desequilíbrio na carga de trabalho não excede d, ou seja, $W_i' \leq (1 + d)(W_{seen} + I) / p$, onde $1 \leq I \leq p$. Como todos os intervalos são processados conforme eles são gerados, diz-se que o algoritmo é puramente online.

O algoritmo Online/Offline é aquele que processa os intervalos ordenados pelo atributo de junção. Este algoritmo é dividido em duas fases. A primeira fase é chamada de online e é a fase em que são armazenados p intervalos de um único valor. O trabalho associado a eles é guardado em uma lista chamada de lista offline. Quando a fase online termina, os intervalos restantes na lista offline são processados pela fase offline. Todos os outros intervalos terão sido processados pela fase online.

A fase online mantém uma lista chamada de lista de agrupamento. Esta lista consiste de intervalos adjacentes que serão agrupados para formar um intervalo multi-valorado. Isto é feito para que o número de intervalos atribuídos aos processadores seja reduzido. O objetivo é obter um critério de balanceamento de carga usando o mínimo de intervalos possíveis. No início da fase online a lista de agrupamento está vazia. No princípio qualquer processador pode ser escolhido. À medida em que as tarefas vão sendo distribuídas, o processador menos carregado é escolhido como sendo o novo processador corrente.

A segunda fase do algoritmo online/offline tem como objetivo balancear a carga de trabalho existente na lista offline (obtida pela fase 1 do algoritmo OO) de forma a minimizar o custo de comunicação (transferência de tuplas) entre os processadores. A heurística utilizada é a seguinte: (i) ordenar os intervalos da lista offline em ordem decrescente de trabalho e processar os dados nesta ordem; (ii) determinar o menor número de divisões necessárias para cada intervalo; (iii) atribuir as partes dos intervalos aos **s** processadores mais lentos. A idéia é preservar os processadores mais rápidos para minimizar futuras divisões de trabalho.

Foram utilizadas duas medidas para verificar o desempenho dos algoritmos: o número de intervalos atribuídos a cada processador e o custo total de comunicação usando fragmentação e replicação quando os intervalos foram divididos. Este custo foi comparado ao custo total do trabalho. Os experimentos com o algoritmo Online Puro mostraram que o número de intervalos gerados aumenta ligeiramente. Apesar de ser obtido um número maior de divisões em OP, dividir um intervalo não aumenta o número de intervalos atribuídos.

2.4.2. Escalonamento e Balanceamento de Carga

O caso geral de um escalonamento online é o seguinte. Tem-se um conjunto de m máquinas e uma determinada sequência de tarefas online que devem ser processadas. Cada tarefa tem um tempo de processamento que pode ou não ser conhecido a priori. Esta tarefa deve ser direcionada imediatamente para uma das m máquinas, sem conhecimento sobre qualquer tarefa futura. O objetivo é otimizar uma dada função, como por exemplo, minimizar o tempo total da última tarefa executada. Existem muitas variações deste problema, podendo ser estudados vários tipos de máquinas e de funções.

Um dos problemas mais básicos de escalonamento considera que as m máquinas são idênticas.

Uma estratégia para tratar este problema é atribuir a nova tarefa à máquina menos carregada. Em [Bar97] é apresentada a prova de que este algoritmo, considerado Guloso, é (2-1/m)-competitivo e também são apresentadas novas técnicas para melhorar este algoritmo online para escalonamento de tarefas.

Em relação ao balanceamento de carga online, tem-se novamente um conjunto de m máquinas e uma sequência de tarefas que chegam em tempo de execução (online). Cada tarefa tem um custo e uma duração que podem ou não ser conhecidos a priori. A qualquer momento a carga de uma máquina é a soma dos custos das tarefas presentes na máquina naquele instante. O objetivo é minimizar a carga máxima que ocorre durante o processamento da sequência de tarefas ? .

É importante observar que, quando a soma de todas as tarefas têm duração infinita, o problema do balanceamento de carga pode ser visto como um problema de escalonamento pois as máquinas estarão sempre processando as tarefas solicitadas.

Assim como no caso do escalonamento, um dos problemas mais básicos de balanceamento de carga considera que as m máquinas são idênticas. Desta forma, tem-se a sequência de tarefas e cada tarefa possui um custo cuja duração é desconhecida. Suponha que carga de uma determinada máquina no instante **t** seja igual a soma dos pesos de todos as tarefas presentes neste instante. O objetivo é minimizar a carga máxima em cada uma das máquinas.

Para o caso em que as máquinas são idênticas, mostrou-se em [ABK92] que o algoritmo Guloso é (2-1/m)-competitivo [Bar97]. O balanceamento de carga torna-se mais complicado no caso onde cada tarefa pode apenas ser atribuída a um subconjunto de máquinas [Aza98].

2.4.3. Comparação de Estratégias Paralelas de Junções

Diversas estratégias de junção paralela foram propostas na literatura. Após a premissa de uniformidade dos dados ter sido posta em dúvida [SD89][LY90], foram desenvolvidas muitas estratégias dotadas de mecanismos de equilíbrio de carga.

Para que essas estratégias pudessem ser comparadas, foram desenvolvidas diversas implementações de ambientes paralelos de avaliação. Enquanto algumas foram desenvolvidas com o objetivo de se comparar uma determinada estratégia nova com outras [DNS+92][LT94], outras foram construídas para comparar estratégias já existentes [SD89][HTY95].

O trabalho apresentado em [SD89] foi o primeiro a unificar em um único ambiente de hardware e software os algoritmos mais usados até o momento. Esse mesmo trabalho mostra que há a necessidade de se balancear carga nos processadores, pois as relações podem gerar cargas de trabalho diferentes para cada processador, fenômeno posteriormente definido como um desvio intrínseco aos dados.

Muitos ambientes de testes foram desenvolvidos usando hardware e software diferentes uns dos outros, o que dificulta a comparação dos trabalhos [SD89][WLH98]. Enquanto alguns utilizam hardware com características específicas e/ou de difícil acesso [WLH98], outros usam softwares especiais, voltados para uma determinada aplicação ou ambiente de desenvolvimento.

Em [HTY95] o problema de se comparar diversas estratégias implementadas em ambientes diferentes foi também abordado. Esse trabalho implementou algumas das estratégias mais usadas na época em uma máquina com hardware paralelo específico.

Devido às diferentes condições de hardware e software, torna-se difícil obter um consenso em relação a forma de se efetuar a comparação entre as diferentes estratégias de execução de junção paralela propostas na literatura. Além disso, cada proposta de comparação define uma metodologia diferente considerando para isso uma lista de parâmetros a serem medidos [DNS+92][LT94][SD89][HTY95].

Como proposta de comparação mais recente, tem-se o artigo [CRS+99] que define uma lista de fatores a serem medidos para que se possa avaliar por completo a performance de uma junção paralela: tempo de computação (tempo total gasto no processamento); tempo de comunicação (tempo gasto enviando e recebendo mensagens); tempo de execução (tempo total gasto na execução completa); volume de comunicação (quantidade e tamanho das mensagens transferidas); volume de E/S (número de operações de E/S); média de paralelismo (número médio de processadores ocupados, fazendo cálculos ou comunicando) e perfil de execução (número de processadores ocupados ao longo do

tempo). Na verdade, trata-se apenas de uma explicitação dos parâmetros a considerar e de como considerá-los.

2.5. Conclusões

Este capítulo apresentou o contexto de estratégias de junção paralela tratado nesta dissertação e apresentou como motivação a idéia inicialmente proposta em [LLP99] de utilizar técnicas de análise de algoritmos online para avaliar o comportamento das estratégias de execução da junção em paralelo. Baseado nos resultados experimentais que podem vir a ser obtidos, os otimizadores dos SGBDs poderiam utilizar estas informações para definir o melhor plano de execução para uma consulta, definindo qual estratégia de execução paralela deveria ser utilizada na presença de um determinado tipo de adversário.

Para explicar melhor o contexto dos algoritmos online, foram apresentados os principais conceitos da teoria online e da análise competitiva, dando como exemplo o problema de paginação. Finalmente, foram descritos alguns trabalhos relacionados ao tratamento online de desvios na junção paralela, algoritmos online de escalonamento de tarefas e balanceamento de carga, e técnicas de comparação de estratégias de junção paralela sugeridas na literatura.

Após terem sido definidas as idéias básicas a serem tratadas nesta dissertação, o próximo capítulo deverá apresentar uma proposta onde será aplicada a análise competitiva para avaliar o desempenho de uma determinada estratégia de execução paralela com balanceamento de carga.

Capítulo 3

Uma Metodologia de Avaliação da Junção Paralela

Esta dissertação propõe comparar o comportamento de uma determinada estratégia online de junção paralela em relação ao comportamento de uma estratégia (offline) ótima, estratégia de execução onde a distribuição da carga de trabalho é a mais equilibrada possível. Como adversário para realizar esta análise de competitividade entre as estratégias online e offline serão utilizados adversários de dados, ou seja, diferentes distribuições de valores nos atributos de junção. Além disso, o número total de processadores disponíveis identifica mais precisamente os adversários considerados.

Visando apresentar uma metodologia de comparação, este capítulo tem como objetivo caracterizar o problema da junção paralela sob o ponto de vista online. Em seguida serão descritos a arquitetura VAST-PJ e o modelo de custo proposto para definir o tamanha de cada uma das tarefas. Serão definidos ainda possíveis tipos de adversários para o problema da junção paralela. Finalmente, é proposta uma métrica para realizar a comparação entre uma determinada estratégia online de junção paralela e a sua contrapartida offline.

3.1. Caracterizando a Junção Paralela como um Problema Online

A operação de junção tem sido uma das mais estudadas em SGBDs, particularmente no contexto de sistemas de memória distribuída [SD89][HTL99][LL98]. Um dos problemas enfrentados na execução de junções paralelas é a distribuição de carga de trabalho pelos processadores participantes [SD89]. O objetivo principal da junção paralela é alcançar um tempo de execução mínimo e a ocupação máxima dos nós processadores.

Esta operação pode ser vista como um conjunto de múltiplas tarefas a serem executadas, onde cada tarefa, por exemplo, corresponde à comparação de diferentes subconjuntos de tuplas das duas relações. Deste modo, a junção de cada tupla pode ser processada de forma independente uma da outra. Uma das formas mais conhecidas para gerar o conjunto de tarefas é através da separação de tuplas de acordo com os valores do atributo de junção [SD89].

Após a construção inicial do conjunto de tarefas a ser processado, a estratégia de execução da junção paralela deve atribuir cada uma das tarefas, eventualmente agrupadas, aos processadores disponíveis de acordo com um determinado critério de distribuição de tarefas por ela definido.

A característica online da operação de junção paralela origina-se do fato de que a divisão e o agrupamento de tarefas deve ser feito sem um conhecimento exato de todos os dados necessários para gerar um conjunto equilibrado de tarefas.

É importante notar que, na prática, um grande número de trabalhos trata o problema de junção de forma online. Isto pode ser observado pelo fato dos algoritmos primeiro particionarem as relações de entrada tentando, a seguir, dividir as tarefas entre os processadores. Esta abordagem frequentemente obtém sucesso no balanceamento da cardinalidade das tarefas alocadas (trabalho associado ao número de comparações de tuplas) principalmente porque a cardinalidade é geralmente conhecida nesta fase. Na maioria das vezes estes algoritmos não consideram o trabalho associado à geração de tuplas (gravação da relação resposta à consulta no disco), pois a seletividade do resultado e a carga de aplicações externas são frequentemente desconhecidas pelos algoritmos no momento em que o particionamento das tarefas é realizado.

Sendo assim, um algoritmo online para junção paralela com balanceamento de carga deverá tentar construir e atribuir cada uma das tarefas a um determinado processador, usando todas as informações sobre as relações de entrada que possam interessar, e tomará decisões sobre o balanceamento de carga baseadas nestas informações parciais. Por outro lado, um algoritmo offline de balanceamento de carga de junção paralela é aquele que gera o conjunto de tarefas com o conhecimento exato de cada um dos parâmetros relevantes para obter a melhor distribuição da carga de trabalho entre os processadores.

No caso do algoritmo online, a falta de informações para a construção do conjunto de tarefas a serem processadas provém principalmente [LLP99]:

(a) do fato de que a seletividade da junção com um grau aceitável de precisão continua em aberto, apesar de já ter sido estudada com detalhe no passado [Ram00][EN00];

(b) do possível conhecimento incompleto do conjunto inicial de tarefas definido pela estratégia utilizada; e

(c) da existência de cargas devido à execução de aplicações externas no processador quando o mesmo está executando uma tarefa.

Para todas estas situações, um algoritmo online encontrará dificuldades em tomar decisões para dividir ou agrupar tarefas, lidando assim com soluções certamente não-ótimas.

3.1.1. Falta de conhecimento do Fator de Seletividade

Para desenvolver funções de custo para a operação de junção com razoável exatidão é preciso ter uma estimativa do tamanho (cardinalidade ou número de tuplas) da relação gerada após a execução da operação [EN00]. Isto é geralmente mantido como uma taxa do tamanho da relação resultante da junção sobre o tamanho do produto cartesiano. Este valor é denominado fator de seletividade da junção (**Sj**).

Sendo |R| o número de tuplas de uma relação R, tem-se que

$$Sj = |(R \; X_c \; S)| \; / \; |(R \; X \; S)| = |(R \; X_c \; S)| \; / \; |R| \; * \; |S|$$

Se não há condição **c** para a junção, então **sj = 1** e a junção é o mesmo que o produto cartesiano. Por outro lado, se nenhuma tupla da relação satisfaz à condição de junção, então **sj = 0**. Em geral, **0 ? sj ? 1**. Para uma junção onde a condição **c** é uma igualdade **R.A = S.B**, tem-se os seguintes casos especiais:

1. Se A é uma chave de R, então $|(R \; X_c \; S)| \; ? \; |S|$, logo $sj \; ? \; 1 \; / \; |R|$
2. Se B é uma chave de S, então $|(R \; X_c \; S)| \; ? \; |R|$, logo $sj \; ? \; 1 \; / \; |S|$

Como calcular o fator de seletividade exato é quase tão difícil quanto resolver o problema de junção, os otimizadores de consulta procuram determinar um valor estimado para a seletividade da junção [EN00]. Um algoritmo online tentará balancear a carga contando apenas com as informações parciais disponíveis.

Um algoritmo online de seleção de tarefas deve gerar um conjunto de tarefas e atribuir cada tarefa **j** a um processador conhecendo apenas o valor máximo para o fator de

seletividade da tarefa, os limites inferior e superior de tempo para a execução da tarefa e os resultados das tarefas anteriores de junção.

É importante destacar que vários trabalhos relativamente recentes obtiveram progressos na estimativa do fator de seletividade [FMS96][GGM+96][SS94]. Os resultados destes trabalhos podem ser usados para estimar mais realisticamente valores para a estimativa da seletividade máxima da junção trazendo provavelmente mais exatidão às decisões tomadas pelos algoritmos online.

3.1.2. Falta de Conhecimento da Sequência de Tarefas

A fim de fazer uma divisão e uma combinação ótimas de tarefas, um algoritmo deve conhecer o conjunto completo de tarefas a serem realizadas a priori. Uma outra característica online é a junção de duas relações cujos dados de entrada são apresentados ao algoritmo de forma parcial. Este é o caso se considerarmos que operadores sobre um plano de execução são frequentemente parte de um *pipeline* de dados. Um algoritmo de junção pode receber informações parciais sobre os dados relacionados às operações iniciadas anteriormente. Estas operações, apesar de produzirem tuplas de saídas, podem não ter finalizado ainda seu processo.

3.1.3. Carga de Aplicações Externas

Os algoritmos de junção podem ser, eventualmente, executados em máquinas paralelas concorrendo com aplicações externas ou mesmo outras aplicações sobre o SGBD. Desta forma, um terceiro problema online é o fato de que o algoritmo deve tomar decisões de balanceamento sem o conhecimento exato da carga externa que poderá estar sendo executada no momento do processamento da tarefa submetida.

Se, no momento em que o processador estiver executando uma tarefa, uma carga externa for submetida ao mesmo processador, a tarefa não é mais executada com exclusividade e, portanto, aumenta o tempo exato de execução da tarefa. A possibilidade de que ocorra uma carga externa em um dado instante no processador é imprevisível. Pode-se apenas obter uma série histórica do passado e assim efetuar-se algumas previsões de carga que podem ou não ser verdadeiras.

Um algoritmo online para distribuição de tarefas deve gerar e atribuir cada tarefa aos processadores conhecendo apenas a estimativa do tempo de processamento da tarefa em modo exclusivo, a série histórica de carga externa submetida a cada processador e o resultado das tarefas anteriores de junção.

É interessante notar que os três casos acima são independentes, ou seja, eles podem acontecer ao mesmo tempo, acrescentando complexidade a uma análise de um algoritmo que trate os casos combinados. Cabe observar que apenas o problema da falta de conhecimento sobre a seletividade das tarefas geradas será analisado nesta dissertação.

Nem todas as estratégias de junção paralela podem ser consideradas online. Porém, para realizarmos um estudo de caso é necessário escolher uma estratégia online de execução paralela com balanceamento de carga, e assim, optou-se pela utilização da arquitetura VAST-PJ descrita a seguir.

3.2. Arquitetura VAST-PJ

Em [LL98] foi proposta uma forma para executar a junção paralela com balanceamento de carga. Ao invés de um algoritmo, VAST-PJ (*Variable Sized Tasks – Parallel Join*) é uma arquitetura cujas principais características estão no tratamento preventivo do balanceamento de carga e na variação do tamanho da tarefa demandada a cada nó.

Resumidamente, a VAST-PJ se desdobra em três componentes básicos: Gerente de Execução, Gerente de Tarefas e Junção Local. O primeiro é o componente controlador da junção paralela e é executado no nó coordenador. É o responsável por escolher quais *buckets* serão processados por cada nó, de maneira que nenhum nó fique sobrecarregado. O Gerente de Execução pode escolher um ou mais *buckets* para formar a tarefa do nó requisitante, ou ainda comandar a quebra de um *bucket* em partes menores. O Gerente de Tarefas é o coordenador local nos nós processadores. É o responsável por enviar as tuplas dos *buckets* que serão processados em outro nó e receber as que serão processadas localmente. Por fim, a Junção Local é a responsável por processar os *buckets* escolhidos pelo Gerente de Execução. A Junção Local faz uso de um SGBD residente em cada nó para processar os *buckets* da melhor forma possível.

A diferença entre o VAST-PJ utilizado neste estudo de caso e a implementação realizada em [Fre00] e [Mac00] é basicamente a forma de estimar o tamanho das tarefas que

estarão sendo distribuídas aos processadores. Estes trabalhos calculam o tamanho da tarefa como sendo o número de buckets que deverão ser processados por uma determinada tarefa. Em [Ler98] também é proposta uma forma diferente para estimar o tamanho da tarefa que leva em consideração o número de páginas lidas e escritas, a estimativa de seletividade inicial disponível para a arquitetura VAST-PJ, entre outros.

A próxima seção apresenta um modelo de custo que será usado para definir o tamanho das tarefas distribuídas aos processadores conforme as tarefas anteriores vão sendo executadas.

3.3. Modelo de Custo para Definir o Tamanho da Tarefa

O particionamento é uma técnica usada na operação de junção paralela através da qual as tuplas das relações envolvidas na junção são reagrupadas em *buckets*. O agrupamento das tuplas nos *buckets* baseia-se no valor de *hash* do atributo da junção. Desta forma, a operação de junção pode ser tratada como uma subjunção de cada *bucket*, permitindo que as subjunções sejam executadas em paralelo.

O modelo de custo apresentado nesta seção é baseado nesta técnica de execução de junção. Esta escolha foi motivada pelo fato de existirem trabalhos de pesquisa anteriores [Ler98][Fre00][Mac00] no Departamento de Informática da PUC-Rio que têm estudado diversos aspectos deste tipo de estratégia e também pelo fato de ter sido desenvolvida uma Arquitetura de Comparação de Junção Paralela (ARCOJP) [Fre00], que permite a implementação de uma estratégia de alocação de tarefas por demanda baseada em [LL98], permitindo a definição dinâmica do tamanho das tarefas a serem alocadas pelos processadores.

Considere agora que o tamanho da tarefa será definido como sendo o trabalho associado à execução da mesma e tem-se que W_t denota o trabalho total associado a execução da junção completa.

Para que o balanceamento de carga seja perfeito, W_t deve ser dividido de forma igual para todos os p processadores envolvidos na execução da junção. Logo, o caso ideal será aquele em que o trabalho atribuído a cada processador for $W_p = W_t / p$.

Define-se ainda que o trabalho associado a um *bucket* das relações operandas (W_b) será uma função do (i) número de tuplas lidas; (ii) número de tuplas resultantes; (iii) número

de comparações efetuadas; e (iv) número de tuplas recebidas para completar o *bucket* da junção.

A seguir são listados alguns parâmetros para a definição do trabalho associado a um determinado *bucket* decorrente da junção de R com S:

- **n** denota a cardinalidade do *bucket* de R;
- **m** denota a cardinalidade do *bucket* de S;
- **r** denota o número de tuplas resultantes da relação;
- W_L denota o trabalho associado à leitura de uma tupla;
- W_E denota o trabalho para se escrever uma tupla;
- W_c denota o trabalho de comparar duas tuplas;
- W_{co} denota o trabalho de receber a comunicação de mensagem de uma tupla.

A partir das definições acima, descreve-se W_b como sendo o somatório do trabalho associado à leitura das tuplas com o trabalho de comparar as tuplas de R e S do *bucket*, mais o trabalho de escrever as tuplas resultantes do par de *bucket*, mais o trabalho associado a receber as tuplas dos outros nós e escrevê-las no disco. É importante ressaltar que as tuplas dos *buckets* serão distribuídas de forma igual para cada um dos nós envolvidos na execução da junção. Cada nó possui **n/p** e **m/p** tuplas de cada *bucket* de R e S gerados inicialmente pelo algoritmo. Devido a esta distribuição inicial de dados, cada nó deverá receber o número total de tuplas do *bucket* (**n+m**) menos o número de tuplas armazenadas em seu disco **[(n+m)/p]**.

Baseada nas definições acima, propõe-se aqui a função de trabalho associada a um *bucket* como sendo:

$$W_b = (n+m)*W_L + r*W_E + (n*m)*W_c + [(n+m)-(n+m)/p](W_E + W_{co})$$

Desta forma, o trabalho associado a um determinado *bucket* será a soma do trabalho de ler as tuplas de R e de S, mais o trabalho de escrever as tuplas resultantes no disco, mais o custo de comparar as tuplas de R com as tuplas de S, mais o custo de receber e escrever no disco as tuplas do *bucket* que estão localizadas nos outros nós.

O tamanho da tarefa para a implementação de uma estratégia de junção paralela com balanceamento nesta dissertação é definido como sendo o trabalho associado à execução de um determinado conjunto de um ou mais *buckets* (W_t).

O problema deste cálculo de trabalho é definir o número de tuplas resultantes r do *bucket*, pois este valor só é conhecido após a execução da junção do *bucket*. Assim, tornou-se necessário definir uma forma para estimar a seletividade do par de *buckets* a ser processado.

Pelo fato desta dissertação ter por princípio que o balanceamento de carga da junção paralela é tipicamente online, considerou-se que o fator de seletividade estimado da junção completa é desconhecido, e desta forma, a única fonte de informação conhecida a cada passo é o resultado das tarefas realizadas anteriormente.

Inicialmente, estima-se que o fator de seletividade da junção é 1. Na medida em que as tarefas vão sendo executadas e os resultados são enviados para o nó central, o coordenador da junção calcula a seletividade das tarefas realizadas e considera que a seletividade da junção completa será igual à seletividade global do passado FS_p.

Desta forma, não apenas a estratégia de balanceamento de carga da junção paralela é considerada como sendo online, como também a estimativa da seletividade das tarefas a serem realizadas também é realizada de forma online, já que o algoritmo estima as seletividades futuras baseadas apenas nas informações dos resultados obtidos no passado, sem nenhuma informação em relação às seletividades das próximas tarefas a serem solicitadas pelos processadores.

Detalhando ainda mais esse processo de definição do tamanho da tarefa, quando as primeiras tarefas são solicitadas pelos nós processadores, o nó coordenador (vide Figura 2) distribui uma tarefa com tamanho inicial aproximadamente igual para cada nó processador, considerando que a seletividade da junção é igual a 1.

Com o passar do tempo, os nós processadores terminam as tarefas iniciais e passam como informação para o nó coordenador os seguintes dados resultantes (Figura 2): (i) número de tuplas resultantes; (ii) tempo de processamento; (iii) número de tuplas recebidas para realizar a tarefa; e (iv) número de tuplas enviadas para os outros nós.

O algoritmo deve a seguir atualizar seu histórico de informações e recalcular a seletividade incremental global da junção até o momento da requisição de tarefa (FS_D). Esta seletividade levará em conta apenas o somatório das comparações efetuadas por todos os *buckets* processados e o número de tuplas resultantes até o momento. Por isso é um cálculo incremental.

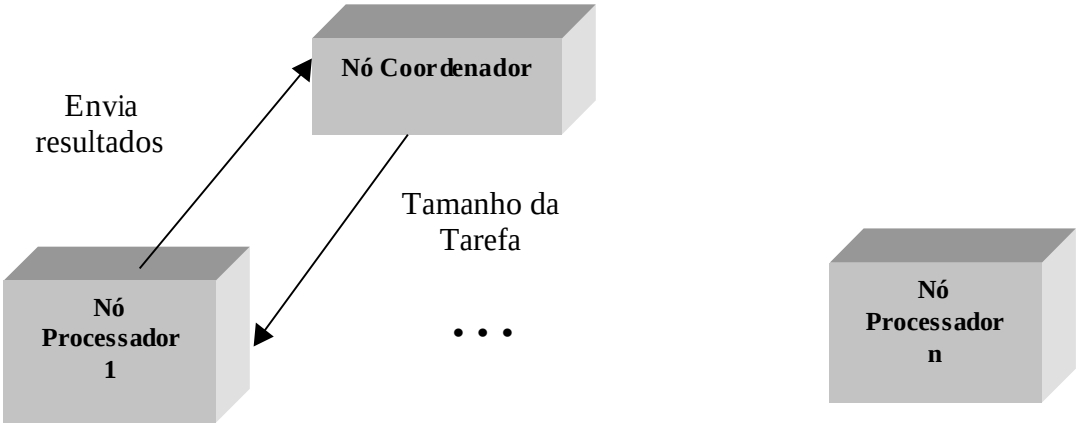

Figura 2: Troca de informações entre nó coordenador e nó processador

A partir do valor de FS_D o nó coordenador estima o valor de r e, consequentemente, um tamanho para a próxima tarefa a ser executada pelo nó solicitante.

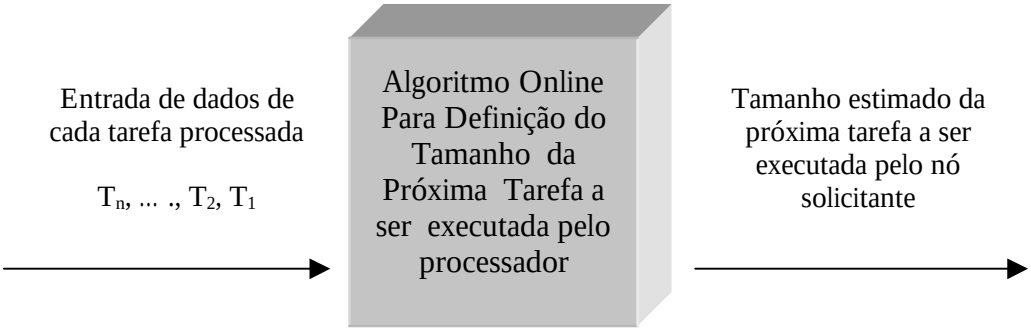

Figura 3: Algoritmo online para determinação do tamanho da próxima tarefa

O tamanho para a próxima tarefa do processador solicitante deverá variar em relação à velocidade de processamento em que a última tarefa foi executada por ele. A velocidade de processamento é calculada como o tamanho da tarefa executada dividido pelo tempo de execução. O tamanho da tarefa varia na mesma proporção que a velocidade variou de uma execução para outra, por exemplo 10% a mais ou a menos para as duas medidas. As informações sobre o tempo de execução de cada processamento de tarefa são obtidas através do agente de software desenvolvido em [Mac00].

Após a caracterização da execução paralela da junção como um problema online e da definição de um modelo de custo para definir o tamanho das tarefas geradas, propõe-se utilizar a análise competitiva para avaliar o comportamento da estratégia online. Como foi definido em 2.3.3, a análise de competitividade é realizada em relação a um adversário. A próxima seção definirá o tipo de adversário utilizado para verificar a competitividade do algoritmo online a ser analisado.

3.4. Tipos de Adversários

A idéia por trás dos adversários de algoritmos de junção é gerar condições que desafiem a capacidade do algoritmo para balancear eficientemente a carga de trabalho entre os processadores. Portanto, os adversários devem ser capazes de reproduzir um grande número de cenários de dados reais (uniformes e não uniformes) de SGBDs uniformes e não-uniformes que os algoritmos devem estar preparados para lidar.

Diferentes condições de não-uniformidade podem ser obtidas através de variações na distribuição dos dados e das características da plataforma às quais o algoritmo deve se adaptar.

Adversários para distribuição não-uniforme de dados estão principalmente preocupados em produzir diferentes tipos de desvios. A não-uniformidade pode ser modelada através de diferentes funções. As distribuições de dados foram também amplamente usadas na avaliação de algoritmos de junção.

As distribuições utilizadas nesta dissertação foram sintetizadas usando a fórmula baseada na Lei de Zipf [Zip49]:

$$|V_i| = \frac{|R|}{i^{Z_p} \cdot \sum_{j=1}^{N} \frac{1}{j^{Z_p}}}$$

onde N é o número de valores distintos e Z_p é um fator de distribuição. Quando $Z_p = 1$, a distribuição dos dados segue a Lei de Zipf (uma distribuição não uniforme próxima da função x^{-1}) e quando $Z_p = 0$, a distribuição é uniforme. A escolha da Lei de Zipf para definir possíveis adversários de dados se deve ao fato de que estes tipos de distribuição de dados são os mais encontrados na prática.

Em relação às características da plataforma, deve-se conhecer a habilidade de um adversário para impor desafios a um algoritmo baseada nos parâmetros da plataforma, principalmente a heterogeneidade e a concorrência. A heterogeneidade pode ser usada por um adversário para configurar uma máquina paralela com diferentes capacidades dos processadores. A concorrência pode ser usada para reduzir a capacidade computacional do processador instantaneamente, como se a carga de uma aplicação externa estivesse usando parte da disponibilidade do processador.

Em relação aos adversários, esta dissertação tem por objetivo avaliar o comportamento de uma determinada estratégia online (VAST-PJ [Ler98]) em relação a diferentes distribuições de dados e o número de processadores disponíveis. Trata-se de um caso de adversário óbvio , como definido na seção 2.3.4.

O problema neste momento é definir qual seria a distribuição ótima da carga de trabalho entre os processadores. Como trata-se de um problema offline, pode-se considerar que todos os fatos relevantes para equilibrar a carga de trabalho são previamente conhecidos pelo algoritmo ótimo.

Baseado nesta idéia de que o algoritmo offline possui um conhecimento completo das informações, o Apêndice deste trabalho apresenta uma proposta de heurística para definir a distribuição offline (ótima) da carga de trabalho.

É importante observar que quando o problema de equilíbrio da carga de trabalho é definido como no Apêndice, ele pode ser considerado como sendo o problema [SS8] descrito em [GJ79]. Desta forma, as heurísticas propostas para tratar este problema podem também ser aplicadas no contexto desta dissertação.

Neste momento, falta ainda definir quais fatores deverão ser coletados para realizar a análise competitiva do algoritmo online de junção paralela com balanceamento de carga avaliado. Ou seja, deve-se definir uma métrica para efetuar a comparação entre os algoritmos online e offline.

3.5. Fatores para Comparação de Estratégias

A análise competitiva é baseada na diferença da qualidade de soluções geradas pelos algoritmos online e offline (ótimo) [Alb97]. Assim, um dos mais importantes aspectos

neste tipo de análise é a definição de uma métrica. Nesta seção, discute-se como obter medidas mais precisas da qualidade de saída e apresenta-se uma métrica, sugerida em [LLP99].

Deve-se notar que obter uma solução perfeita de balanceamento nem sempre garante o melhor tempo de resposta. Por exemplo, se o trabalho realizado para equilibrar a carga for muito grande em relação ao trabalho de se executar a junção paralela, torna-se melhor obter uma solução com um desequilíbrio na distribuição da carga do que obter a solução ótima.

O artigo [LLP99] propôs três parâmetros de análise para definir uma métrica para quantificar a qualidade da solução resultante de um algoritmo de junção paralela com balanceamento de carga. São eles: Razão de Comparação de Tupla (número de tuplas na relação resultado sobre o número de comparações efetuadas), Diferença de Carga de Trabalho (mede a maior diferença entre a carga de trabalho de dois processadores) e Taxa de Transferência de Tuplas (número de vezes que as tuplas são lidas e escritas).

Nesta dissertação a carga de trabalho será avaliada de forma um pouco diferente. Pretende-se analisar:

1. a taxa de comparação de tuplas;

2. a taxa de leitura de tuplas; e

3. a taxa de tuplas resultantes (tuplas da relação resultado).

Estes parâmetros foram escolhidos pois são considerados na literatura as operações mais importantes do ponto de vista do processamento de tarefas.

A partir destes três valores serão calculadas as diferenças entre as taxas encontradas nas estratégias online e offline. Estes parâmetros são detalhados a seguir.

Os principais parâmetros a serem coletados são as taxas de tuplas lidas e escritas no processamento da tarefa. Estes valores são obtidos dividindo-se o número de tuplas lidas ou escritas por processador pelo número total de tuplas lidas ou escritas por todos os processadores. Desta forma, as medidas apresentadas para análise representarão uma porcentagem do número total de tuplas lidas ou escritas por processador.

Quanto à taxa de comparação de tuplas, deve ser obtido por processador o número de comparações realizadas em relação ao número total de comparações necessárias para se executar a junção. Para cada um dos três parâmetros coletados, deve-se calcular a diferença máxima entre o valor experimental obtido e o valor encontrado pela estratégia offline.

Apesar da existência de muitos outros parâmetros úteis (como por exemplo os parâmetros propostos em [LLP99]), a combinação destes três parâmetros já é suficiente para ilustrar a efetividade de um algoritmo. Para cada estratégia, os valores serão influenciados pela configuração do problema como por exemplo, frequência de valores sobre o atributo de junção, número de processadores, etc.

3.6. Conclusões

Este capítulo teve por objetivo caracterizar a junção paralela como um problema online, apresentando como fatores online de análise a falta de conhecimento quanto à seletividade da junção, quanto a sequência de tarefas a serem executadas e quanto à carga de aplicações externas que podem ser submetidas aos processadores no momento em que a junção paralela está sendo executada.

Sugeriu-se então, utilizar a estratégia VAST-PJ para ilustrar o uso da análise competitiva para avaliar o comportamento de uma determinada estratégia de execução de junção paralela com balanceamento em relação aos adversários de dados, considerados importantes devido a sua ligação com o problema da falta de conhecimento da seletividade da junção.

Por ser uma análise competitiva, a distribuição considerada ótima (offline) para a carga de trabalho será a divisão igual para cada um dos processadores disponíveis. Desta forma, a distribuição obtida pelas estratégias online serão comparadas com a distribuição perfeita da carga de trabalho.

Finalmente, foram definidos três parâmetros de comparação – taxa de leitura, escrita e comparação de tuplas – que deverão ser coletados das execuções das estratégias online e offline, acrescentando-se ainda as diferenças encontradas entre as versões online e offline.

O próximo capítulo apresentará um estudo de caso, utilizando a estratégia VAST-PJ, para mostrar a aplicabilidade da análise competitiva no contexto de comparação de estratégias de junções paralelas.

Capítulo 4
Estudo de Caso

Neste capítulo pretende-se realizar uma avaliação da abordagem de comparação online para analisar, na prática, a competitividade de uma estratégia de execução da junção paralela com balanceamento de carga. Para isso, utiliza-se a métrica definida no Capítulo 3 para avaliar a distribuição da carga de trabalho total entre os processadores envolvidos na execução da junção.

A estratégia de junção paralela com balanceamento de carga utilizada aqui é uma implementação da arquitetura VAST-PJ [LL98] cujas características principais são o tratamento preventivo do balanceamento de carga e a variação do tamanho da tarefa demandada a cada nó. O que se costuma encontrar na literatura é o balanceamento corretivo, onde a estratégia detecta o desequilíbrio na distribuição de carga e depois tenta reequilibrar. A versão da VAST-PJ nesta dissertação utilizou o cálculo estimado para o tamanho da tarefa definido no Capítulo 3.

A implementação apresentada usou uma arquitetura para comparação de junções paralelas denominada ARCOJP [Fre00]. Esta arquitetura utiliza componentes de um protótipo de SGBD relacional (Minibase)[Ram97] para realizar as operações básicas de um SGBD. A escolha desta arquitetura foi motivada pelo fato da ARCOJP permitir o uso da VAST-PJ, configurando-se o gerente de tarefas segundo o modelo de custo proposto nesta dissertação.

É importante ressaltar que nesta análise não estaremos levando em consideração o tempo de execução das junção paralela. Serão analisados apenas os parâmetros que estão relacionados à carga de trabalho como leitura, escrita e comparação de tuplas. Pretende-se examinar a eficácia da VAST-PJ quanto ao balanceamento de carga.

As seções a seguir descrevem brevemente a arquitetura ARCO-JP e o ambiente de desenvolvimento onde serão realizados os testes. Em seguida é apresentada a aplicação da heurística proposta no ApêndiceApêndiceApêndiceApêndice ApêndiceApêndiceApêndicepara mostrar que os resultados ótimos (offline) obtidos são

aproximadamente iguais à divisão perfeita entre os processadores disponíveis. Por fim são apresentados alguns dados obtidos experimentalmente e a análise destes resultados.

4.1. Arquitetura ARCOJP

A ARCOJP (ARquitetura para COmparação de Junções Paralelas) é motivada pelo trabalho [LL98] onde foi proposta uma arquitetura para execução da junção paralela com tarefas de tamanho variável visando balanceamento de carga preventivo denominada VAST-PJ.

Foi definida uma arquitetura mais genérica que a VAST-PJ, a ARCOJP, para permitir a execução de diversos algoritmos de junção paralela, usando os componentes básicos definidos pela arquitetura VAST-PJ. A arquitetura estendida objetivou a criação de um ambiente para comparação de algoritmos paralelos de junção que pudesse ser facilmente replicado. Esse objetivo foi alcançado através da utilização de componentes de hardware (computadores do tipo PCs e *cluster*) e software (sistema operacional Linux, linguagem C++, mecanismo de comunicação MPI) padrões e da criação de componentes de alto nível para implementação de novos algoritmos.

A arquitetura ARCOJP foi construída usando o conceito de arquitetura em camadas. As suas camadas principais são (Figura 4): comunicador, gerente de dados, nós processadores e componentes de junção paralela. Essas camadas isolam e flexibilizam os aspectos de comunicação e armazenamento de dados, permitindo que qualquer mecanismo de armazenamento e comunicação seja utilizado. Esta nova arquitetura é detalhada e validada em [Fre00], onde são apresentados resultados comparativos de quatro algoritmos paralelos de junção implementados usando a arquitetura. Em [Mac00], esta arquitetura foi usada para realizar um estudo sobre a extensão de componentes utilizando agentes de software.

A Figura 4 apresenta as camadas principais da arquitetura ARCOJP. Os componentes Gerente de Execução, Gerente de Tarefa e Gerente de Junção Local são responsáveis pela execução dos algoritmos paralelos de junção.

O objetivo da construção em camadas foi tornar a arquitetura independente dos aspectos de comunicação e armazenamento de dados, permitindo a implementação de algoritmos paralelos de junção através do uso de componentes básicos da arquitetura. A subcamada

Nós Processadores serviu para representar internamente a topologia externa dos nós processadores. Desta forma, na implementação do algoritmo, o programador se preocupa somente com o envio e recebimento de mensagens, sem necessidade de saber detalhes sobre a comunicação [Mac00].

Figura 4: Camadas principais da arquitetura ARCOJP

É importante ressaltar que a arquitetura ARCOJP serviu como alicerce não só para a implementação das idéias discutidas neste trabalho, como também para a captura das informações sobre a distribuição da carga de trabalho que serão analisadas.

A ARCOJP possui duas formas de coletar informações sobre a execução da junção paralela. A primeira é um arquivo texto onde são armazenadas as informações estatísticas como tempo de execução, quantidade de bytes transmitidos, quantidade de *bytes* lidos e gravados, etc. A segunda forma de se capturar informações é o arquivo de log onde é descrita grande parte dos passos para a execução da estratégia como por exemplo, transmissões de mensagem, cardinalidade dos *buckets* e o número de tuplas resultantes por *bucket*.

O componente fundamental para esta dissertação foi o Gerente de Tarefas (GT), que é o componente responsável pelo cálculo do tamanho das tarefas a serem atribuídas aos processadores. Como a interface entre o GT e os outros componentes é um número representando o tamanho da tarefa a ser executada e uma lista dos *buckets* que compõem

a tarefa, foi necessário alterar apenas o GT, visto que a interface de comunicação entre os componentes existentes continuou a mesma.

4.2. Ambiente de desenvolvimento

Nesta seção serão descritos os componentes de hardware e software utilizados na implementação da estratégia online de execução de junção paralela com balanceamento de carga e os dados do ambiente de teste.

4.2.1. Hardware, Software e Comunicação

O hardware utilizado para o desenvolvimento da arquitetura foi um *cluster* formado por 32 processadores Pentium II 400 MHz IBM PC 300 GL, cada um com unidade de disco própria, conectados por uma rede Ethernet através de um switch IBM 8274 modelo W93 na placa ESM-C-32W que permite 32 segmentos a 10 Mbps.

Os computadores componentes do *cluster* funcionaram sob o sistema operacional Linux Red Hat 5.2 (Apollo), com kernel versão 2.0.36. A linguagem utilizada na implementação foi C++, o compilador foi o gcc versão GNU v2.9 e o depurador gdb.

A comunicação entre os nós processadores da máquina paralela foi realizada utilizando um software de comunicação que implementa o protocolo denominado MPI (*Message Passing Interface*) [GL96][GL97]. A implementação utilizada foi o LAM-MPI, atualmente na versão 6.3b [LAM96].

4.2.2. Condições do Ambiente de Teste

Esta seção descreve as condições do ambiente de teste ao qual a estratégia será submetida.

Baseado no fato de que a maioria das junções é feita entre um atributo chave primária e sua correspondente chave estrangeira, decidiu-se por seguir o modelo apresentado em [HTL99, HTY95]. Este modelo define duas relações R e S, sendo a cardinalidade de R menor que a de S. A relação R possui uma coluna C0 que é uma chave primária e a relação S possui, entre outras, uma coluna C0 que é uma chave estrangeira referente a coluna C0 da tabela R. Com isso, o objetivo será realizar a junção entre as tabelas R e S com a cláusula de junção R.C0 = S.C0.

Para o caso da base de teste, tem-se a relação menor R que contém uma coluna C0 (chave primária), possuindo desta forma 50.000 valores distintos. Existe ainda a relação S que possui, nas colunas C0 a C10, os 50.000 valores possíveis para a chave primária de R, com o fator Z_p variando de 0.1 para cada atributo. Assim, a coluna C10 possui uma distribuição de Zipf e a coluna C0 possui uma distribuição uniforme.

As relações R e S possuem outras colunas com o objetivo apenas de configurar um tamanho considerável para o tamanho das tuplas

Resumidamente, a base de testes é composta pelas relações R e S, descritas na Figura 5.

Características	*Relações*	
	R	*S*
Tamanho:	50.000 tuplas	100.000 tuplas
Atributos:	C0 – Inteiro	C0 – C10 – Inteiro
Distribuição dos valores:	C0 – 50.000 valores diferentes	C0 a C10 distribuídos segundo uma variação da função Zipf

Figura 5: Características das Relações de teste

A Figura 6 mostra a distribuição inicial de tuplas pelos *buckets*, para alguns valores de Z_p, após a geração da base de dados.

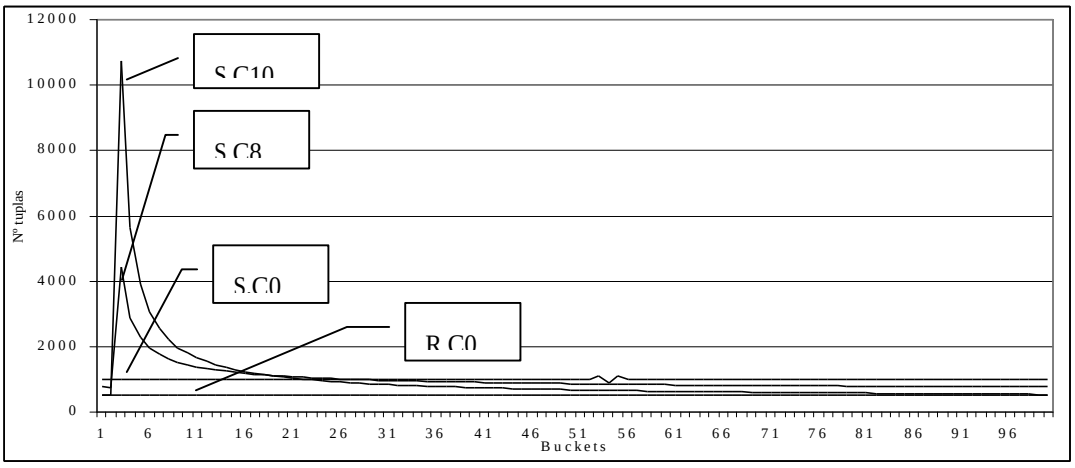

Figura 6: Distribuição inicial de tuplas

Como pode ser visto, os atributos C0 de R e S estão uniformemente distribuídos. Conforme o fator Z_p tende a 1, a distribuição perde a uniformidade até chegar a uma distribuição de Zipf completa (atributo C10 de S).

Os outros dados da base de teste estão descritos na Figura 7. Basicamente foram mantidas as mesmas condições da base de teste utilizadas em [Fre00] e [Mac00].

Observe que o fato da distribuição inicial dos dados ser circular influenciou no modelo de custo para estimar o tamanho das tarefas definidos no Capítulo 3, pois leva-se em conta o fato de cada processador possuir, para cada *bucket* armazenado no seu disco, uma cardinalidade aproximadamente igual ao número de tuplas dividido pelo número de processadores que executam a junção paralela.

A ARCOJP permite que estas configurações sejam alteradas, porém optou-se por manter esses dados para que fosse possível efetuar posteriormente outros tipos de comparações.

Tamanho da base de dados	300MB em cada processador
Modo de distribuição inicial das tuplas	Circular (*round-robin*)
Tuplas por transferência	8
Buffer pool local	2000 páginas
Tamanho da página	1KB
Tamanho máximo da tupla	1KB
Tamanho das tuplas	34 bytes para R 74 bytes para S
Buckets	100
Páginas recuperadas por E/S	1

Figura 7: Características do ambiente de teste

Como foi explicado anteriormente, segundo a teoria de algoritmos online, o conceito de adversários é definido como sendo um algoritmo que seja capaz de simular diferentes situações do mundo real para verificar o comportamento de um dado algoritmo online que está sendo analisado. Desta forma, as diferentes distribuições de dados às quais a estratégia de execução de junção paralela será submetida podem ser chamadas de adversários de dados (adversário óbvio) do algoritmo que está sendo analisado.

Nesta dissertação serão utilizados basicamente três diferentes adversários de dados (distribuições): distribuição uniforme, distribuição Zipf com fator $Z_p = 0.5$ e distribuição Zipf com fator $Z_p = 1$. Além dos diferentes adversários de dados, variou-se o número de processadores que executam a junção, sendo usados 2, 4 e 8 processadores.

Os resultados experimentais obtidos serão comparados com a distribuição ótima da carga de trabalho definida como sendo a divisão perfeita da carga de trabalho entre os processadores disponíveis.

4.3. Determinação da distribuição ótima da carga de trabalho (offline)

Apesar da comparação dos dados experimentais (online) ser feita em relação à divisão perfeita da carga de trabalho, esta seção descreve como foram obtidas as distribuições ótimas (offline) para as bases de teste.

Baseado na heurística descrita no Apêndice deste trabalho, foi realizado o cálculo de uma distribuição considerada ótima de acordo com as condições às quais o algoritmo online de junção paralela está sendo submetido. Em todos os casos, a situação ótima encontrada foi aquela em que todos os processadores executam a mesma carga de trabalho em relação à leitura, escrita e comparação de tuplas. Por exemplo, se a junção paralela for executada por 2 nós, a situação ótima será aquela em que cada nó processador executa 50% de cada uma das operações da carga de trabalho.

Para chegar a este resultado foram necessários vários procedimentos. O primeiro passo da heurística consiste em efetuar uma análise da seletividade das colunas da tabela S em relação às colunas da tabela R. Pelo fato do ambiente do Minibase não possuir uma interface amigável para a manipulação dos dados nem possibilitar a exportação destes

dados para outro aplicativo, surgiu a necessidade de importar as tabelas R e S para um ambiente mais amigável que possibilitasse a análise destas tabelas.

Como os dados foram importados para as tabelas do Minibase via arquivo texto, foi gerado um programa (em Java) que lê o conteúdo deste arquivo e constrói um outro arquivo de saída, cujos dados estão formatados de forma que o SGBD SQL Server 7.0 possa importar para dentro de suas tabelas, armazenadas no ambiente Windows NT.

Após a importação das tabelas de teste para o SQL Server, percebeu-se a necessidade de obter uma forma prática de gerar as tabelas iniciais de análise necessárias para a execução da heurística. Devido a isso, as tabelas do SQL Server foram vinculadas dentro de um banco de dados Microsoft Access 97 para que os resultados obtidos pudessem ser facilmente transferidos para o Excel em forma de tabelas. A importação dos dados não foi efetuada diretamente pelo Access 97 pelo fato deste aplicativo Windows não ter suportado a importação de 100.000 tuplas de uma só vez.

Observe ainda que devido ao fato das colunas C0, C5 e C10 serem chaves estrangeiras em relação a coluna C0 da tabela R, a análise da tabela S se resume a fazer uma contagem, para cada uma das três colunas em questão, de quantas tuplas existem em S por valor existente. Esta análise foi obtida através de uma consulta cujos valores de seleção eram a coluna de S que está sendo analisada e um contador, agrupando o resultado pela coluna em questão. Por exemplo, para a coluna C0 foi submetida a consulta "**`select C0, count(*) from S group by C0`**".

Após a obtenção das tabelas iniciais de análise, era necessário aplicar a heurística para distribuir a carga de trabalho entre 2, 4 e 8 nós. Este procedimento foi realizado para cada um dos adversários de dados sugeridos para análise. Para facilitar, aplicou-se as técnicas descritas pela heurística para equilibrar a carga entre dois processadores e obteve-se uma distribuição da carga de trabalho de aproximadamente 50% para todos os três adversários propostos. Isso ocorreu pelo fato das tabela iniciais de análise possuírem um número bastante variado para a coluna que representa o número de tuplas de S que irão se combinar com cada uma das tuplas de R.

Para calcular a distribuição para quatro processadores, dividiu-se em duas cada uma das duas tarefas geradas na etapa anterior. Os resultados obtidos foram aproximadamente 25% da carga de trabalho para cada nó. Analogamente, para o caso de oito

processadores, dividiu-se em duas cada uma das quatro tarefas geradas, como na fase anterior. O resultado obtido foi aproximadamente 12,5% da carga de trabalho para cada nó.

Baseado nestes resultados obtidos, a análise será feita em comparação a uma distribuição perfeita de carga entre os processadores tanto para leitura, escrita e comparação de tuplas.

4.4. Resultados Experimentais

Nesta análise, entende-se por carga de trabalho todo o trabalho necessário para ler as tuplas da junção, comparar os atributos de junção e escrever as tuplas no disco.

É importante observar que pode acontecer de dois nós estarem com a mesma carga de trabalho, porém um executa mais comparações de valores e o outro escreve mais tuplas na relação resultado. Isto se deve ao fato de que cada uma das operações está associada a um determinado custo em relação às outras. Note que as operações de entrada e saída são mais caras que a operação de comparação de tuplas.

Nas seções a seguir serão apresentados separadamente os valores medidos para cada um dos processadores que executam a junção para que se tenha uma idéia melhor sobre qual operação cada processador executou. Os gráficos apresentados comparam os seguintes valores:

1. Porcentagem de leitura de tuplas em relação ao total (%L)

   %L = n.º de tuplas lidas pelo nó **`i`** / n.º de tuplas lidas por todos os nós

2. Porcentagem de tuplas resultantes em relação ao total (%R.); e

   %R = n.º de tuplas escritas pelo nó **`i`** / n.º de tuplas escritas por todos os nós

3. Porcentagem de comparações em relação ao total (% C).

   %C = n.º de comparações no nó **`i`** / n.º de comparações por todos os nós

4. Porcentagem ótima da carga total de trabalho. De acordo com a heurística apresentada no Apêndice, a distribuição ótima da carga de trabalho é

aproximadamente a divisão perfeita entre os processadores para as operações de leitura, escrita e comparação de tuplas. Sendo assim, a distribuição ótima será apresentada por apenas uma coluna nos gráficos, porém representando as três operações que estão sendo analisadas.

%O = carga total de trabalho (leitura, escrita ou comparação)/ n.º de nós processadores

5. % DifC é a maior diferença entre a taxa de comparação de tuplas %C e a distribuição ótima %O. Analogamente, % DifL é a maior diferença entre a taxa de leitura de tuplas %C e a distribuição ótima %O e % DifR é a maior diferença entre a taxa de escrita de tuplas %C e a distribuição ótima %O.

Os dados necessários para a análise de resultados foram capturados através dos arquivos textos gerados pela ARCOJP. Os arquivos guardam todas as informações consideradas relevantes pela arquitetura, como por exemplo, tempo de execução, cardinalidade dos *buckets* e número de tuplas resultantes, como também são informados todos os passos executados pelos nós processadores e pelo nó coordenador para que a junção seja finalizada.

É importante ressaltar que de acordo com a seção anterior, a distribuição da carga de trabalho ótima para 2, 4 e 8 processadores será 50%, 25% e 12,5% respectivamente.

4.4.1. Adversário - Distribuição Uniforme

A distribuição da carga de trabalho total entre os nós processadores para a distribuição uniforme de dados para 2, 4 e 8 processadores é apresentada pelos gráficos das 24, 25 e 26 respectivamente.

Observe que quando a distribuição de dados é uniforme, a distribuição da carga de trabalho é bem próxima da distribuição ótima para as três operações analisadas. Por exemplo, veja a Figura 8 onde a diferença entre os parâmetros %L, %C e %R é de apenas 1% em relação ao ótimo. Note que esta variação de 1% entre a estratégia online se manteve mais ou menos constante em relação ao número de processadores.

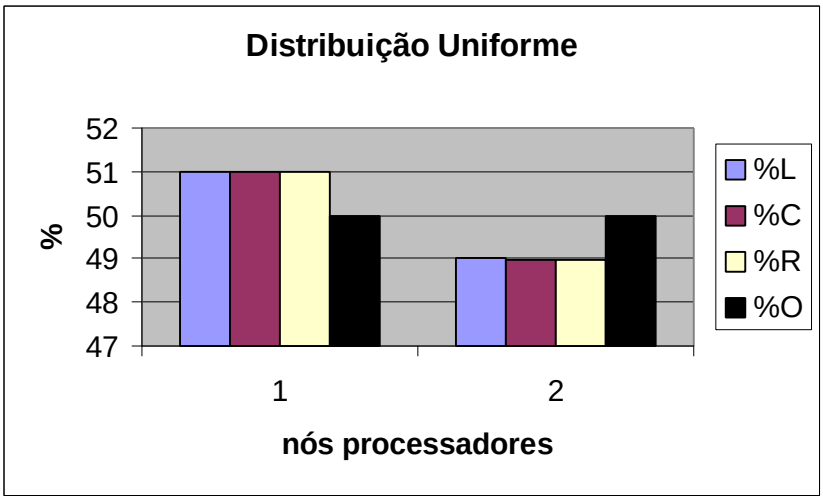

Figura 8: Distribuição da carga de trabalho entre 4 nós, dados uniformes

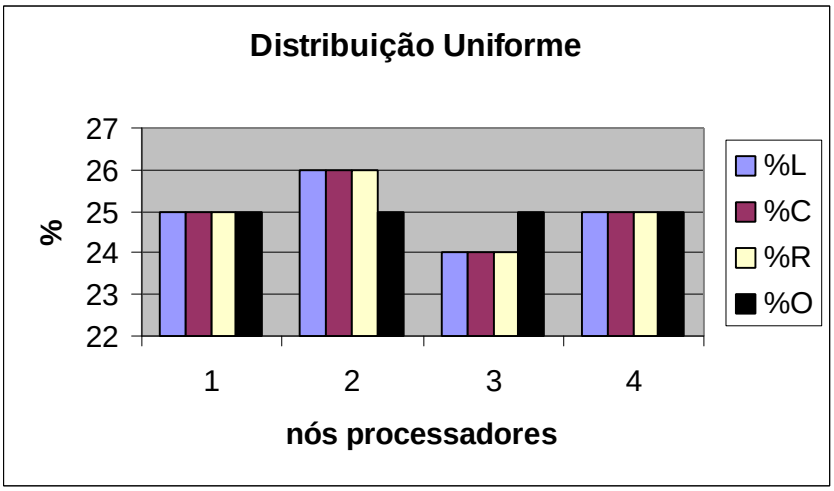

Figura 9: Distribuição da carga de trabalho entre 4 nós, dados uniformes

O fato da distribuição da carga na estratégia online estar bem próxima da distribuição ótima não quer dizer que o tempo de execução do VAST-PJ e do algoritmo ótimo serão os mesmos, pois para o caso ótimo existe apenas uma divisão inicial ótima da carga de trabalho sem a necessidade dos processadores solicitarem tarefas através de troca de mensagens com o nó coordenador, além da VAST-PJ executar uma série de outros procedimentos como por exemplo estimar o tamanho da próxima tarefa a ser enviada para um determinado nó.

A partir da análise dos gráficos, pode-se obter ainda os valores máximos de diferença entre a carga de trabalho para os itens medidos, ilustrado no gráfico da Figura 11. Esta medida foi obtida de acordo com o número de nós processadores envolvidos na execução da junção.

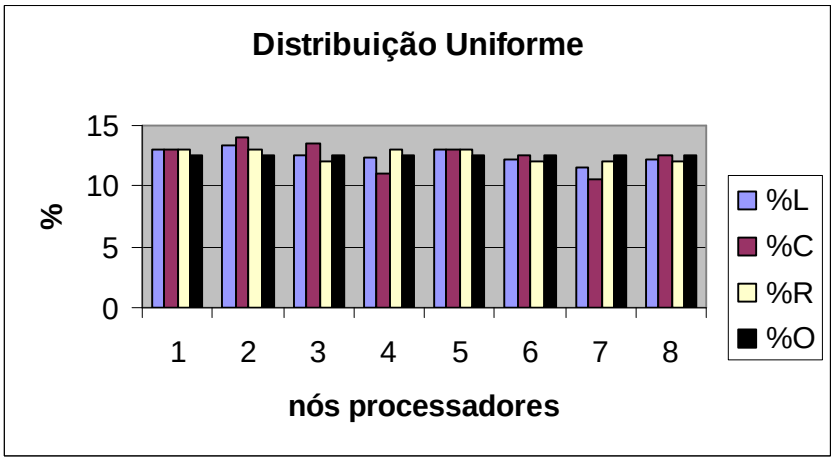

Figura 10: Distribuição da carga de trabalho entre 8 nós, dados uniformes

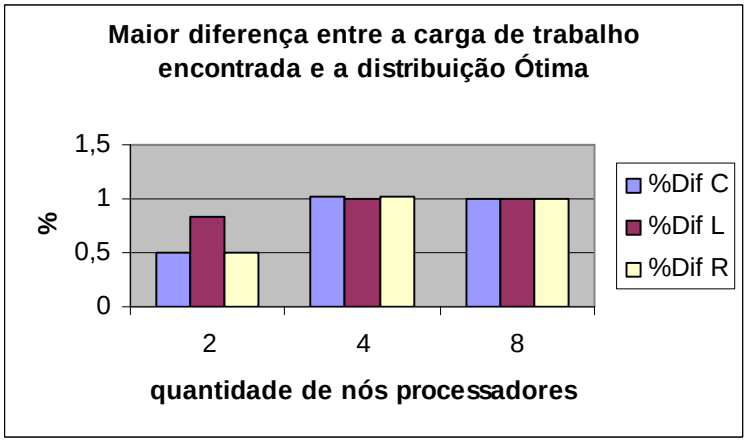

Figura 11: Diferença máxima na carga de trabalho para distribuição uniforme

Observe que a diferença entre a carga de trabalho de todas as operações não ultrapassou 2% tanto para a diferença de comparação de tuplas quanto para a diferença máxima entre o número de tuplas resultantes. Isto ocorreu devido à distribuição dos dados ser uniforme. Neste caso, os *buckets* de R e S gerados pela estratégia implementada possuem basicamente a mesma cardinalidade e a mesma seletividade. Desta forma, o número de

tuplas lidas, escritas e comparadas é praticamente o mesmo. Estes resultados se aproximam do caso ótimo, onde a diferença na carga de trabalho é igual a zero.

4.4.2. Distribuição Zipf fator 0.5

A distribuição das carga de trabalho entre os nós processadores para a distribuição Zipf com fator Z_p = 0.5 entre 2, 4 e 8 processadores é representada pelas figuras 29, 30 e 31.

Neste caso, a distribuição da carga de trabalho é mais irregular do que para o adversário de distribuição uniforme de dados. Observe que à medida em que o número de processadores aumenta, a distribuição da carga de trabalho fica mais distante da distribuição ótima. Observe que para 2 processadores, a diferença entre a distribuição online e a ótima (offline) é de cerca de 3%, porém quando o número de processadores aumenta para 8, esta diferença se aproxima de 6%.

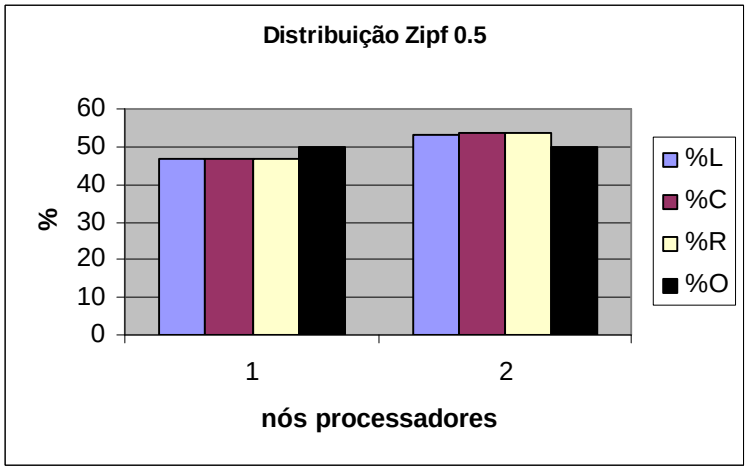

Figura 12: Distribuição da carga de trabalho entre 2 nós, dados Zipf 0.5

As distribuições fornecidas pelos gráficos do adversário de dados Zipf 0.5 permitem que sejam obtidos os valores máximos de diferença entre a carga de trabalho para cada um dos itens medidos, possibilitando assim a construção do gráfico da Figura 15. Estas medidas foram obtidas em relação ao número de nós processadores envolvidos na execução da junção.

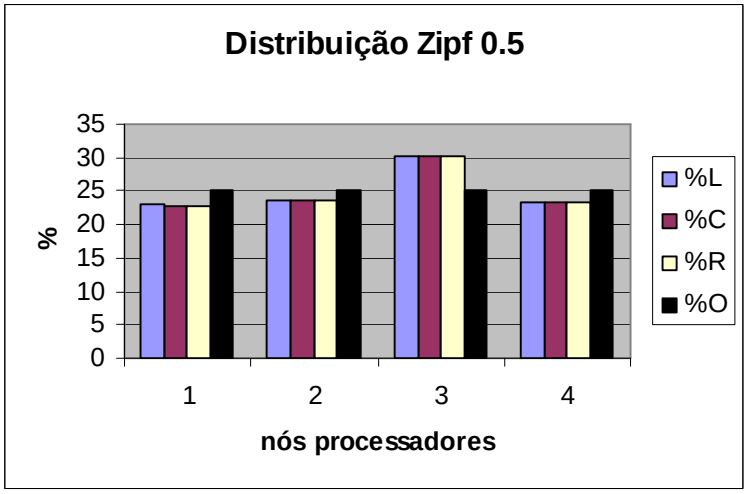

Figura 13: Distribuição da carga de trabalho entre 5 nós, dados Zipf 0.5

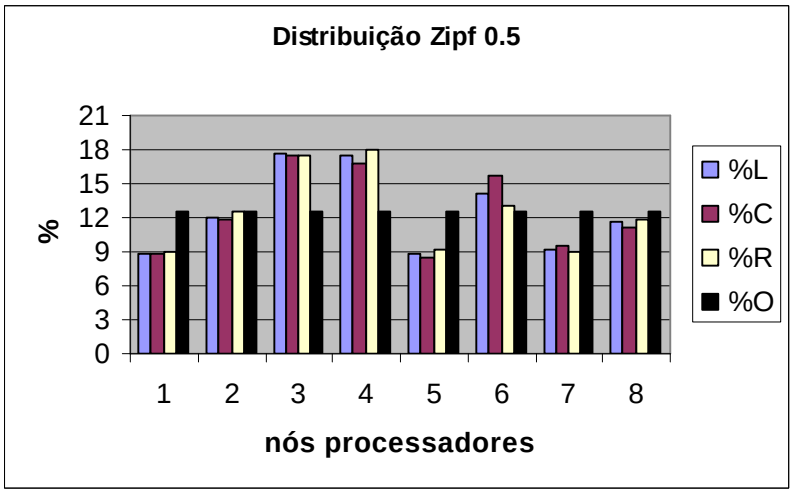

Figura 14: Distribuição da carga de trabalho entre 8 nós, dados Zipf 0.5

A diferença entre a carga de trabalho de todas as operações atingiu um valor máximo de 5.05% para a comparação de tuplas, 5.1% para o número de tuplas lidas e 5.2% para o número de tuplas resultantes. Como a distribuição dos dados é Zipf 0.5, já existe um desvio nos nós, o que significa que alguns *buckets* das relações R e S possuem uma maior cardinalidade e possivelmente um maior número de tuplas resultantes. Consequentemente, quando um nó processador recebe estes *buckets*, sua taxa de comparação de tupla aumenta em relação ao total o que pode acarretar um número maior de leitura de tuplas e um aumento do número de possíveis tuplas resultantes.

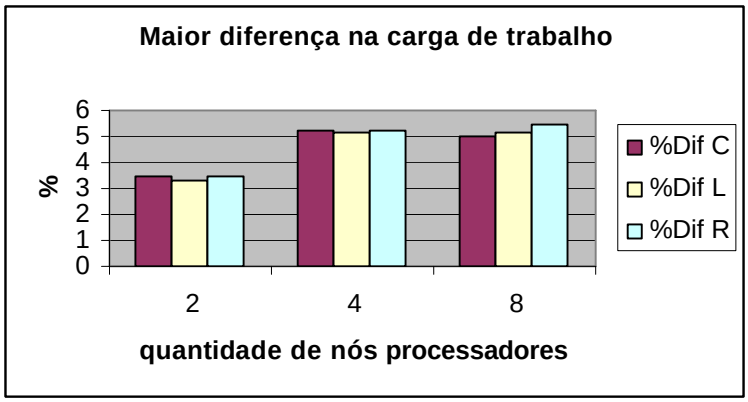

Figura 15: Diferença máxima na carga de trabalho para distribuição Zipf

4.4.3. Distribuição Zipf

A distribuição das cargas de trabalho entre 2, 4 e 8 processadores para a distribuição Zipf completa é apresentado pelas figuras 32, 33 e 34.

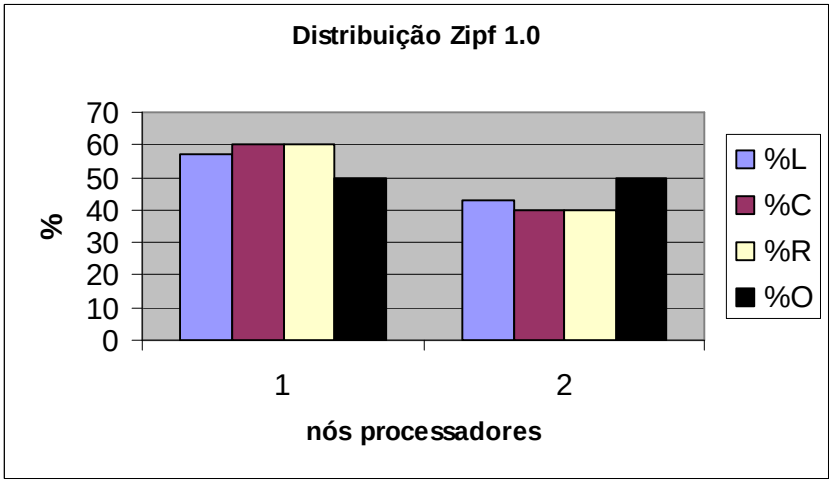

Figura 16: Distribuição da carga de trabalho entre 2 nós, dados Zipf

Observe, por exemplo na Figura 16, que na presença deste tipo de adversário, a diferença entre a distribuição ótima e a distribuição da estratégia online é maior (aproximadamente 10% para leitura de tuplas) que a diferença encontrada na análise dos outros adversários.

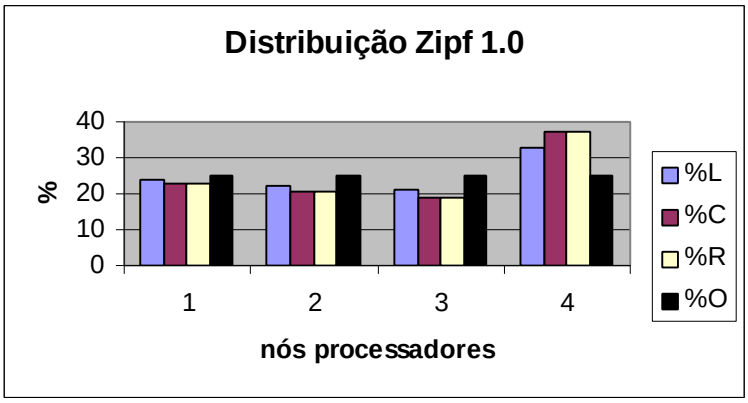

Figura 17: Distribuição da carga de trabalho entre 4 nós, dados Zipf

O gráfico da Figura 19 apresenta a maior diferença na carga de trabalho referente a cada uma das operações realizadas. Esta medida foi obtida de acordo com o número de processadores envolvidos na execução da junção.

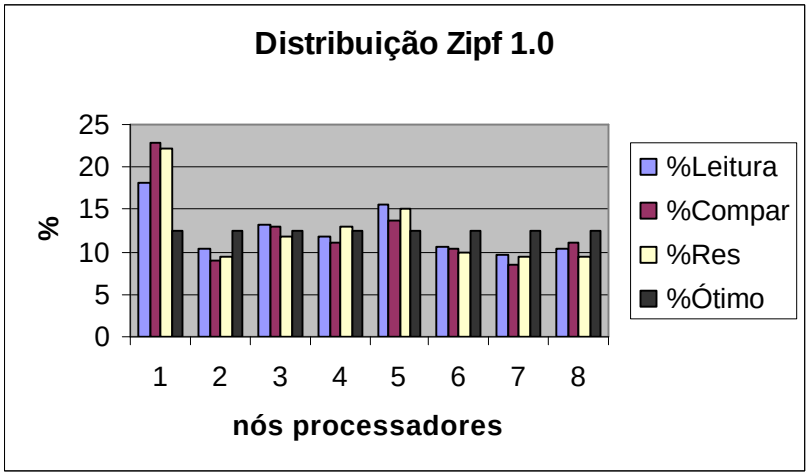

Figura 18: Distribuição da carga de trabalho entre 8 nós, dados Zipf

A diferença entre a carga de trabalho de todas as operações atingiu um valor máximo de 12% para a comparação de tuplas, 12% para o número de tuplas lidas e 7% para o número de tuplas resultantes. Como a distribuição dos dados é Zipf completa, a maioria dos dados está concentrada em uma determinada faixa de valor, o que significa que

poucos *buckets* possuem o maior número de tuplas. Consequentemente, quando um nó processador recebe um destes *buckets*, sua taxa de comparação de tupla aumenta em relação ao total, o que causa um número maior de leitura de tuplas, aumentando o número de possíveis tuplas resultantes.

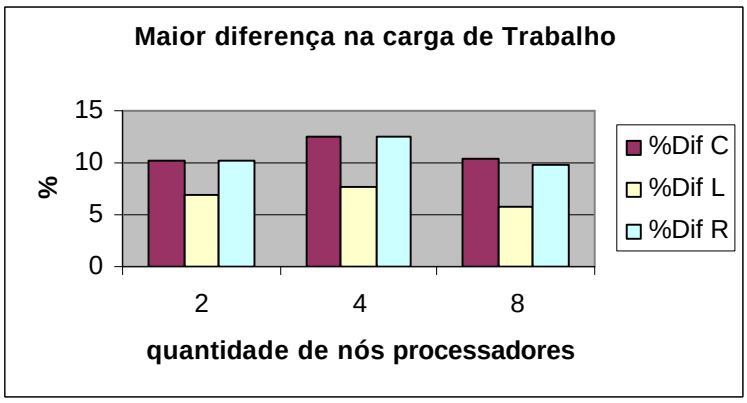

Figura 19: Diferença máxima na carga de trabalho para distribuição Zipf

4.5. Discussão dos Resultados Obtidos

Após a apresentação dos resultados experimentais obtidos, é importante levantar algumas discussões sobre os dados coletados.

O primeiro ponto a ser destacado é que não foi avaliada nesta dissertação a eficiência (tempo de execução) das estratégias online e offline descritas aqui. O objetivo foi realizar uma análise competitiva entre as estratégia online e offline avaliando a distribuição da carga de trabalho associada a cada uma das execuções de junção paralela, ou seja, a eficácia quanto ao balanceamento de carga.

O objetivo desta dissertação é mostrar que a análise competitiva pode ser usada para comparar as estratégias online de junção paralela com uma estratégia offline (ótima) e, assim, a partir dos resultados experimentais, coletar informações que ajudem o otimizador de SGBDs a decidir qual o melhor plano de execução, dado que um determinado adversário está presente na execução da operação.

Ressalta-se ainda que não apenas o adversário de dados foi utilizado para realizar a análise, como também o adversário em relação ao número de processadores disponíveis

para executar a junção. O comportamento observado não foi totalmente uniforme em relação ao número de processadores.

Não foi um objetivo desta dissertação determinar a competitividade da arquitetura VAST-PJ de forma analítica mas sim, comparar experimentalmente as diferenças de carga entre as estratégias online e offline (ótima). Entretanto, a partir dos dados experimentais, pode-se ter uma idéia da competitividade da estratégia online analisada, avaliando os desvios de carga encontrados de acordo com cada adversário proposto.

No caso do adversário que constrói as tuplas de maneira uniforme, a diferença entre a distribuição ótima da carga de trabalho e a distribuição da VAST-PJ não ultrapassou 1% para todas as operações, mesmo variando o número de processadores. Isto não significa que as duas estratégias executam a junção paralela no mesmo tempo. É de se esperar que a VAST-PJ execute a mesma junção em maior tempo pois ela distribui as tarefas em várias etapas, diferente do que ocorre na estratégia offline, onde as tarefas são distribuídas em uma só etapa.

Por outro lado, quando o adversário de dados utiliza a distribuição Zipf 0.5, a diferença entre a distribuição ótima e a VAST-PJ atingiu um máximo de 6%, o que ocorre com o maior número de processadores disponíveis (8). Como era esperado, a diferença máxima neste caso foi maior que a diferença na presença do adversário com distribuição uniforme. Não foi observado uma diferença significativa no valor da diferença máxima quando o número de processadores variou.

Finalmente, na presença do adversário de dados Zipf completo, se confirmou a tendência do aumento da diferença máxima entre a distribuição ótima e o equilíbrio de carga proporcionado pela VAST-PJ, que foi, por exemplo, de 12% para a comparação de tuplas na presença de 4 processadores.

4.6. Conclusões

Nesse capítulo, foi analisado o comportamento da estratégia VAST-PJ quando esta é submetida a diferentes adversários de dados (distribuição uniforme, Zipf 0.5 e Zipf), variando o número de processadores disponíveis. As medidas obtidas experimentalmente representam uma porcentagem do trabalho total referente à comparação, leitura e escrita de tuplas realizado em cada processador. Esta medidas foram comparadas com a

distribuição considerada ótima (offline) definida como sendo a distribuição perfeita da vcarga de trabalho entre os processadores disponíveis. A análise competitiva aqui realizada pode dar uma idéia sobre o comportamento da estratégia de execução de junção paralela online, sem utilizar as medidas de tempo.

A análise competitiva sugerida nesta dissertação pode não ser a melhor forma de avaliar o desempenho das estratégias pelo fato de não contar o tempo, porém ela fornece uma visão geral sobre a distribuição de trabalho entre os processadores (o que possivelmente estará refletido na contagem do tempo de processamento), independente da plataforma de execução e da estratégia online analisada.

Capítulo 5
Conclusões, Contribuições e Trabalhos Futuros

Esta dissertação teve como principal objetivo motivar a utilização da análise competitiva para avaliar o comportamento das estratégias paralelas de junção.

Inicialmente, foi descrito o problema de execução da junção paralela e balanceamento de carga, apresentando como motivação a idéia proposta em [LLP99] de utilizar técnicas de análise de algoritmos online para avaliar o comportamento das estratégias existentes. Para isso, caracterizou-se a junção paralela como um problema online, apresentando como fatores online a falta de conhecimento quanto à seletividade da junção, quanto à sequência de tarefas a serem executadas e quanto à carga de aplicações externas que podem ser submetidas aos processadores no momento da execução.

Baseado nos resultados experimentais obtidos, os otimizadores dos SGBDs poderiam utilizar estas informações para definir o melhor plano de execução para uma consulta, definindo qual estratégia de execução paralela deveria ser utilizada na presença de um determinado tipo de adversário.

A arquitetura VAST-PJ foi utilizada para ilustrar o uso da análise competitiva para avaliar o comportamento de uma determinada estratégia online de execução de junção paralela com equilíbrio de carga em relação aos adversários de dados, relacionados com o problema da falta de conhecimento da seletividade da junção.

Por ser necessário obter uma distribuição ótima (offline) para a carga de trabalho, utilizou-se como parâmetro de comparação a situação ideal onde a carga de trabalho é perfeitamente dividida entre os processadores disponíveis para executar a junção. Foram também definidos três parâmetros de comparação – taxa de leitura, escrita e comparação de tuplas – que foram coletados das execuções das estratégias online e offline, observando-se ainda as diferenças encontradas entre elas.

Em relação ao estudo de caso, foram analisados os comportamentos da estratégia VAST-PJ quando esta é submetida a diferentes adversários de dados (distribuição uniforme, Zipf 0.5 e Zipf) e número de processadores disponíveis. As medidas obtidas experimentalmente representam a porcentagem do trabalho total referente à comparação,

leitura e escrita de tuplas realizadas em cada processador. Esta medidas foram comparadas com a distribuição ótima (offline). Diferentes medidas foram coletadas em relação aos vários adversários utilizados.

Algumas contribuições desta dissertação são citadas abaixo:

- a utilização da abordagem online para caracterização das estratégias de junção paralela com balanceamento de carga;

- a elaboração de um modelo de custo adequado para estimar o tamanho da tarefa dinamicamente na implementação da arquitetura VAST-PJ;

- a realização de um estudo de caso prático com uso da análise competitiva para avaliar o comportamento de uma estratégia online de execução da junção paralela, no caso a VAST-PJ.

Dando sequência a essa dissertação, alguns trabalhos futuros são aqui sugeridos:

- efetuar a análise competitiva proposta para avaliar o comportamento de outras estratégias de execução paralela e compará-las;

- analisar outros parâmetros de comparação entre as estratégias online e offline (ótima), como por exemplo os parâmetros propostos em [LLP99];

- comparar eficiência do modelo de custo das tarefas aqui proposto com o modelo proposto em [Fre00] e [Mac00] e também em [Ler98];

- realizar uma análise competitiva específica para os algoritmos que estimam valores para a seletividade da junção em ambiente não paralelo, comparando com as métricas hoje conhecidas.

Referências Bibliográficas

[ABK92] Azar, Y., Broder, A. e Karlin, A.: *On-line load balancing*. Procs. 36th IEEE Symp. On Foundations of Computer Science, pp. 218-225, 1992.

[Alb97] Albers, S.: *Competitive Online Algorithms*. Optima Math. Prog. Society News., pp.1-8 (54), 1997.

[AL97] Albers, S. e Leonardi, S.: *Online Algorithms*. Comm. of the ACM, Survey, pp. 1-6, 1997.

[Aza98] Azar, Y.: *Online load balancing*. The State of Art, editado por A. Fiat e G. Wolginger, Springer Lecture Note of Computer Science (LNCS), pp. 178- 95 (1442), 1998.

[Bar97] Bartal, Y.: *On-line Computation & Network Algorithms*. Lecture 25, pp. 1-3, 1997. http://www.mpi-sb.mpg.de/~albers/

[BLS92] Borodin, A., Linial, N. e Saks, M.: *An optimal on-line algorithm for metrical task systems*. Journal of the ACM, pp. 745 – 763(39), 1992.

[CRS+99] Cremonesi, P., Rosti, E., Serazzi, G. e Smirni, E.: *Performance Evaluation of Parallel Systems*. Parallel Computing 25, pp. 1677-1698, 1999.

[DG92] DeWitt, D.J., e Gray, J.: *Parallel Database Systems – The Future of High Performance Database Systems*. Comm. of the ACM, pp. 85-98 35(6), 1992

[DNS+92] DeWitt, D.J., Naughton, J.F., Schneider, D.A. e Seshadri, S.: *Pratical Skew Handling in Parallel Joins*. Procs. of the 18th VLDB Conf., pp. 27-40, 1992.

[EN00] Elmasri, R. e Navathe, S. B.: *Fundamental of Database Systems*, Addison-Wesley, 2000.

[FKL+91] Fiat, A., Karp, R. M., Luby, M., McGeoch, L. A., Sleator, D. D. e Young, N. E.: *Competitive paging algorithms*. Journal of Algorithms, pp. 685-699 (12), 1991.

[FMS96] Faloutsos, C., Matias, Y. e Silberschatz, A.: *Modeling Skewed Distributions Using Multifractals and the 80-20 Law'.* Procs. Intl. Conf. on VLDB, pp. 307-317, 1996.

[Fre00] Freitas, F.; ARCOJP – *Uma Arquitetura para Comparação de Algoritmos de Junção Paralela*. Dissertação de Mestrado, Departamento de Informática da PUC-Rio, Setembro de 2000.

[GGM+96] Ganguly, S., Gibbons, P.B., Matias, Y., e Silberschatz, A.*: Bifocal Sampling for Skew-Resistant Join Size Estimation*. Procs. ACM SIGMOD Intl. Conf., pp. 271-281, 1996.

[GJ79] Garey, M. R. e Johnson, D. S.: *Computers and Intractability – A Guide to the Theory of NP-Completeness*, pp. 238, W. H. Freeman and Company; 1979.

[GL96] Gropp, w. e Lusk, e.: *User´s Guide for MPI-CH – A portable implementation of MPI*. Technical Report, Argonne National Laboratory, 1996.

[GL97] Gropp, w. e Lusk, e.: *Why are PVM And MPI So Different?* Recent Advances in PVM and MPI, LNCS nº.1332, 1997

[Goe94] Goemans, M.X.: *On-line Algorithms*. Advanced Algorithms, pp. 1-52, 1994. http://www.mpi-sb.mpg.de/~albers/

[HTL99] Hua, K. A., Tavanapong, W. e Lo, Y.L.: *Performance of Load Balancing Techniques for Join Operations in Shared-nothing Database Management Systems*. Journal of Parallel and Distributed Computing 56, pp.17-46, 1999.

[HTY95] Hua, K.A., Tavanapong, W. e Young, H.C.: *A Performance Evaluation of Load Balancing Techniques for Join Operations on Multicomputer*. Database Systems, Procs. of the ICDE, pp. 17-46, 1995.

[LAM96] *MPI Primer – Developing With LAM*. Ohio Supercomputer Center, Ohio State University, 1996.

[Leo98] Leonardi, S.: *On-line network routing. In Online Algorithms*: The State of Art, editado pela A. Fiat e G. Wolginger, Springer LNCS, pp. 242-267(1442),

1998.

[Ler98] Lerner, A.; *Uma Proposta de Arquitetura para o Tratamento Paralelo da Junção com Equilíbrio de Carga em Ambientes com Memória Distribuída*. Dissertação de Mestrado, Departamento de Informática da PUC-Rio, Março de 1998.

[LL98] Lerner, A. e Lifschitz, S.*: A Study of Workload Balancing Techniques on Parallel Join Algorithms*, Procs. of the Intl. Conf. on Parallel and Distributed Processing Techniques and Applications (PDPTA), pp. 966-973, 1998.

[LPR97] Lifschitz, S., Plastino, A. e Ribeiro, C.C.C*: Exploring Load Balancing in Parallel Processing of Recursive Queries*. Procs. Intl. Conf. Parallel Processing (Euro-Par), LNCS 1300, pp. 1125-1129, 1997.

[LT94] Lu, H. e Tan, K.: *Load Balanced Join Processing in Shared Nothing Systems*. Journal of Parallel and Distributed Computing, vol. 23, 1994.

[LLP99] Lerner, A., Lifschitz, S. e Poggi, M.: *An Online Approach for Parallel Join Algoritms Analysis*. Monografia do Departamento de Informática da PUC-Rio, MCC01, 1999.

[LY90] Lakshmi, M.S., e Yu, P.S.: *Efectiveness of Parallel Joins*. IEEE Trans. On Knowledge and Data Engineering, pp. 410-424 2(4), 1990.

[Mac00] Macêdo, J. A. F.; *Um Estudo de SGDB baseado em Agentes*. Dissertação de Mestrado, Departamento de Informática da PUC-Rio; setembro de 2000.

[ME92] Mishra, P. e Eich, M.H.: *Join Processing in Relational Databases*. ACM Comp. Survey, pp. 63-113, 24(1), 1992.

[MMS90] Manasse, M., McGeoch, L. A., e Sleator, D. D.*: Competitive algorithms for server problems*. Journal of Algorithms, pp. 208 - 230 (11), 1990.

[Ram00] Ramakrishnan, R; *Database Management Systems*; McGraw-Hill; 2000.

[Ram97] Ramakrishnan, R.: The Minibase Homepage, 1997.
http://www.cs.wisc.edu/~dbbook/minibase.html

[SD89] Schneider, D. A. e DeWitt, D.J.: *A Performance Evaluation of Four Parallel Join Algorithms in Shared-Nothing Multiprocessor Environment*. Procs. of ACM SIGMOD Intl. Conf., pp. 110-121, 1989.

[Sga98] Sgall, J.: *On-line Scheduling*. The State of Art, editado por A. Fiat e G. Wolginger, Springer LNCS, pp. 198 – 231 (1442), 1998.

[SS94] Swami, A., e Schiefer, K.B.: *On the Estimation of Join Result Size*. Procs. of Intl Conf. on Advances in Database Technologies, pp. 287-300, 1994.

[SY93] Swami, A., e Young, H.C.: *Online Algorithms for Handling Skew in Parallel Join*. Procs. of Intl. Conf. On Parallel Processing, pp. 253-257, 1993.

[WDJ91] Walton, C.B., Dale, A.G. e Jenevein, R.M.: *A Taxonomy and Performance Model of Data Skew Effects in Parallel Join*. Procs. Intl. Conf. on VLDB, pp. 537-548, 1991.

[WLH98] Weon, Y., Lee, S. e Hong, M.: *A Load-Balanced Parallel Join Algorithm on Hypercube*. Parallel and Distributed Processing Techniques and Applications (PDPTA), pp. 941-950, 1998.

[Zip49] Zipf, G. K.: *Human Behavior and the Principle of Least Effort: An Introduction to Human Ecology*. Addison-Wesley, Reading, Mass, 1949.

Apêndice

Heurística para Calcular Distribuição ó tima da Carga de Trabalho

Esta apêndice apresenta uma heurística para calcular uma distribuição ótima da carga de trabalho necessária para executar uma junção entre duas relações. Observe que a carga de trabalho, no contexto desta dissertação, leva em conta o número de tuplas lidas, o número de tuplas escritas e a quantidade de comparações entre tuplas.

Baseado no fato de que a maioria das junções é feita entre um atributo chave primária e sua correspondente chave estrangeira, suponha que existam duas relações R e S. A relação R possui uma coluna C_1 que é uma chave primária. A relação S possui uma coluna C_2 que é uma chave estrangeira da coluna C_1 de R. O objetivo é executar em paralelo a junção $(R \ X_c \ S)$, distribuindo da melhor forma possível as operações de leitura, escrita e comparação de tuplas entre os processadores.

Suponha que a carga de trabalho é distribuída apenas entre **p** processadores disponíveis. Os passos para calcular a distribuição ótima da carga de trabalho são definidos na Figura 20.

Para entender melhor o mecanismo dos ciclos de distribuição suponha que deve-se distribuir entre dois processadores P1 e P2 oito tarefas cujas quantidades de tuplas de S que se combinam com tuplas de R estão em ordem decrescente (10, 9, 8, 7, 6, 5, 4, 1). Como estas tarefas devem ser distribuídas entre dois processadores, cada duas tarefas da lista define um ciclo de distribuição. No primeiro ciclo a tarefa de tamanho 10 será atribuída a P1 e a tarefa de tamanho 9 a P2. No segundo ciclo, como P1 foi privilegiado anteriormente com a tarefa maior, P2 receberá a tarefa de tamanho 8 e P1 receberá a tarefa de tamanho 7, e assim sucessivamente.

Ao final da distribuição, P1 estará com uma tarefa de tamanho 24 (10, 7 , 6, 1) e P2 com uma de tamanho 26 (9, 8 , 5, 4). Este tipo de distribuição não garante um equilíbrio inicial perfeito para a distribuição de carga, porém sugere uma forma prévia de se tentar construir o balanceamento.

1. Definir para cada valor da coluna C_1 de R, quantas tuplas de S possuem o mesmo valor na coluna C_2;

2. Ordenar a tabela resultante do Passo 1 por ordem decrescente da quantidade de tuplas de S que se combinam com cada tupla de R. Em seguida, deve-se atribuir cada uma das linhas da tabela (definidas como tarefas a serem executadas) a um dos nós de forma a tentar equilibrar o número de tarefas de cada processador. Para isso, define-se um ciclo de distribuição de tarefas que começa distribuindo a maior tarefa ao processador 1, a segunda maior tarefa ao processador 2 e assim por diante até o p-ésimo processador disponível. Nesse momento é dada sequência a um sentido inverso, ou seja, do processador p ao processador 1. Uma vez no processador 1, a atribuição de tarefas volta a inverter a sequência até o p-ésimo processador. E assim o ciclo segue até não mais existirem tarefas;

3. Após o Passo 2, deve-se calcular a soma de tuplas resultantes em cada um dos processadores. Após calcular esta soma, se os valores forem diferentes, deve-se obter o valor de carga do processador menos carregado $\mathbf{q_{min}}$. Sendo $\mathbf{q_i}$ a carga de cada um dos processadores, deve-se obter o somatório de ($\mathbf{q_i - q_{min}}$) e dividir o valor encontrado pelo número **p** de processadores. O valor obtido desta fórmula é chamado q_E. Finalmente, tendo em vista que o ideal seria que cada processador recebesse uma carga igual a ($\mathbf{q_{min} + q_E}$), o algoritmo procura equilibrar a carga entre os processadores através da transferência de tarefas dos processadores mais carregados para os menos carregados, procurando atingir o valor ideal de carga. Esta transferência de tarefas pode ser feita quantas vezes forem necessárias até que a distribuição de tuplas resultantes em cada processador seja igual, ou o mais próximo disso;

4. Finalmente, se possível, deve-se equilibrar o número de tuplas lidas de R transferindo, dos processadores mais carregados para os menos carregados, as tarefas em que as tuplas de R não se combinam com qualquer tupla de S.

Figura 20: Heurística para definição da distribuição ótima

Outro caso que serviria para ilustrar a heurística seria distribuir a carga de trabalho entre três processadores. Nesta situação, as tarefas deveriam ser distribuídas na seguinte ordem: (1,2,3), (2,3,1), (3,2,1), (1,3,2), (2,1,3), (3,1,2) e a partir daí voltar para a distribuição inicial (1,2,3) e repetir até atribuir aos processadores todas as tarefas.

Algumas observações são importantes para se entender melhor as idéias propostas pela heurística para o caso de dois processadores:

- A primeira consideração a ser feita é que a distribuição de tarefas propostas pelo Passo 2 é realizada de forma a favorecer, a cada ciclo de distribuição, um dos processadores que está executando a junção paralela para que se obtenha um conjunto inicial de tarefas mais ou menos equilibrado. Além disso, note que após o Passo 2, garante-se que a diferença na carga de leitura de tuplas de R é praticamente a mesma (existirá uma diferença de uma tupla se a cardinalidade for ímpar).

- A tentativa no Passo 3 de equilibrar o número de tuplas escritas por cada processador pode desequilibrar o número de tuplas lidas de R, pois algumas tarefas foram transferidas de um processador para outro. Devido a este fato, o Passo 4 tentará reequilibrar o número de tuplas lidas de R sem alterar o número de tuplas lidas e escritas de S, transferindo, se existirem, as tarefas que não resultam em tuplas de S.

No caso geral, é importante notar que como as colunas de S são chaves estrangeiras de R, o número de tuplas resultantes de S é igual ao número de tuplas lidas de S. Observe também que pode não existir apenas uma forma de obter o equilíbrio perfeito da carga de trabalho pois diferentes escolhas podem ser feitas em cada um dos passos definidos pela heurística, por exemplo, a escolha de tarefas a serem transferidas de um processador para o outro no Passo 3. Da mesma forma, pode não ser possível, de acordo com os passos acima, obter um equilíbrio perfeito da carga de trabalho. Neste caso, a distribuição ótima não será igual à divisão exata das tarefas a serem executadas.

Estendendo a heurística para situações onde as colunas analisadas no Passo 1 não são chaves estrangeiras do atributo de junção, muda-se apenas o fato de que existirão tuplas de S que não irão combinar com qualquer tupla de R.

A seguir serão apresentados alguns exemplos da execução passo a passo desta heurística tendo relações exemplo com diferentes distribuições de dados.

Suponha que existam duas tabelas R e S como descritas na Figura 20. A tabela R tem cardinalidade igual a 20 e a tabela S tem cardinalidade igual a 200. A tabela R possui a coluna C0 como chave primária e os valores existentes variam de 1 a 20. Por outro lado a tabela S possui quatro colunas C0, C5, C10, e C11 que são chaves estrangeiras referentes a coluna C0 da tabela S. A coluna C0 de R possui uma distribuição uniforme entre seu valores, a coluna C5 possui uma distribuição Zipf com fator Z_p = 0.5, a coluna C10 possui distribuição Zipf completa com Z_p = 1. A coluna C11 possui uma distribuição de valores que não segue nenhum padrão específico. Esta distribuição foi criada para que seja possível analisar um caso em que a distribuição da carga de trabalho não é perfeita.

O objetivo é realizar as junções explicitadas a seguir com a linguagem SQL:

```
Junção 1.  SELECT * from R, S WHERE R.C0 = S.C0;

Junção 2.  SELECT * from R, S WHERE R.C0 = S.C5;

Junção 3.  SELECT * from R, S WHERE R.C0 = S.C10;

Junção 4.  SELECT * from R, S WHERE R.C0 = S.C11;
```

Inicialmente deve-se analisar as tabela R e S como descrito no Passo 1. Observa-se, por exemplo, que as distribuições de valores nas colunas C0, C5, C10 e C11 são apresentadas na Figura 21.

Observe ainda o fato de que para a coluna C0, existe o mesmo número de tuplas (10) de S que irão se combinar com cada uma das tuplas existentes em R. Este fato era de se esperar pois a distribuição da coluna C0 de S é uniforme. Em relação às outras distribuições de dados, observe por exemplo que existem 23 ocorrências do valor 1 na coluna C5 da tabela S e que por outro lado, não existe qualquer ocorrência do valor 18 nesta coluna.

Lembrar que como definido na Figura 20 cada linha da tabela anterior pode ser considerada como uma tarefa para a execução da junção paralela.

Valor da Col. C0 de R	Nº de col. C0 de S	Nº de col. C5 de S	Nº de col. C10 de S	Nº de col. C11 de S
1	10	23	50	120
2	10	20	25	3
3	10	16	10	1
4	10	15	7	7
5	10	10	5	9
6	10	10	3	3
7	10	8	1	1
8	10	4	1	3
9	10	0	0	5
10	10	19	48	10
11	10	17	25	0
12	10	16	10	10
13	10	15	5	5
14	10	9	5	5
15	10	10	3	3
16	10	4	1	1
17	10	3	1	1
18	10	0	0	3
19	10	1	0	4
20	10	0	0	6
Total de Tuplas de S	**200**	**200**	**200**	**200**

Figura 21: Tabela de distribuição de dados nas tabelas R e S

Exemplo 1: Calcular distribuição ótima da carga de trabalho para executar a Junção 1

Com a distribuição de dados coletada pela Figura 21, deve-se executar o Passo 2 da heurística que pede que a tabela obtida pelo Passo 1 seja ordenada em função do número de colunas de S que se combinam com os valores da coluna de R. Após a ordenação da tabela, as tarefas geradas devem ser distribuídas entre os dois processadores que executarão a junção. Como foi dito na Capítulo 3, a operação de se atribuir uma tarefa para cada processador é realizada em ciclos, chamados aqui como ciclos de distribuição. A cada ciclo de distribuição, um dos processadores deverá ser privilegiado com a tarefa de maior tamanho (o tamanho da tarefa é medido pelo número de tuplas de S que irão se combinar com uma determinada tupla de R).

Sendo assim, as tarefas deverão ser distribuídas em ciclos da seguinte forma (1,2), (2,1), (1, 2), (2, 1) e assim sucessivamente. Lembre que o objetivo neste passo é já tentar balancear a carga de trabalho no momento em que as tarefas estão sendo distribuídas.

Para a Junção 1, o resultado obtido a partir do Passo 2 é apresentado na Figura 22. Observe que, devido ao fato da distribuição ser uniforme, todas as tuplas de R irão se

combinar com o mesmo número de tuplas de S. A tentativa de se privilegiar a cada ciclo um dos processadores não seria necessária neste caso pois todas as tarefas possuem o mesmo custo. Para concluir o Passo 2, deve-se agrupar as tarefas distribuídas por processador e calcular o custo associado a cada uma delas como mostrado na Figura 23.

Observe que, a não ser que o número de tuplas de R não seja divisível pelo número de nós processadores (no caso, apenas 2), a distribuição da carga de trabalho será exatamente igual para leitura, escrita e comparação de tuplas. Neste caso, não é necessário executar os passos 3 e 4 pois foi obtida uma distribuição perfeita da carga de trabalho de acordo com a Figura 24.

Os gráficos apresentados comparam o seguintes valores:

1. Porcentagem de leitura de tuplas em relação ao total (%L)

   %L = n.º de tuplas lidas pelo nó **`i`** / n.º de tuplas lidas por todos os nós

2. Porcentagem de tuplas resultantes em relação ao total (%R.); e

   %R = n.º de tuplas escritas pelo nó **`i`** / n.º de tuplas escritas por todos os nós

3. Porcentagem de comparações em relação ao total (% C).

   %C = n.º de comparações no nó **`i`** / n.º de comparações por todos os nós

Valor da Col. C0 de R	Nº de col. C0 de S	Processador que irá executar
1	10	1
2	10	2
3	10	2
4	10	1
5	10	1
6	10	2
7	10	2
8	10	1
9	10	1
10	10	2
11	10	2
12	10	1
13	10	1
14	10	2
15	10	2
16	10	1
17	10	1
18	10	2
19	10	2
20	10	1

Figura 22: Tabela com distribuição inicial da Junção 1

Tuplas de R	Valor da Coluna C0 de R	Nº de col. C0 de S	Processador que irá executar
1	1	10	1
1	4	10	1
1	5	10	1
1	8	10	1
1	9	10	1
1	12	10	1
1	13	10	1
1	16	10	1
1	17	10	1
1	20	10	1
10		100	
1	2	10	2
1	3	10	2
1	6	10	2
1	7	10	2
1	10	10	2
1	11	10	2
1	14	10	2
1	15	10	2
1	18	10	2
1	19	10	2
10		100	

Figura 23: Tabela com distribuição ótima da Junção 1

NP	Tuplas Lidas de R	Tuplas Lidas de S	Total Lida	Comparações	Tuplas Result	%Compar	%Leitura	%Result
1	10	100	110	1000	100	50	50	50
2	10	100	110	1000	100	50	50	50
			220	2000	200			

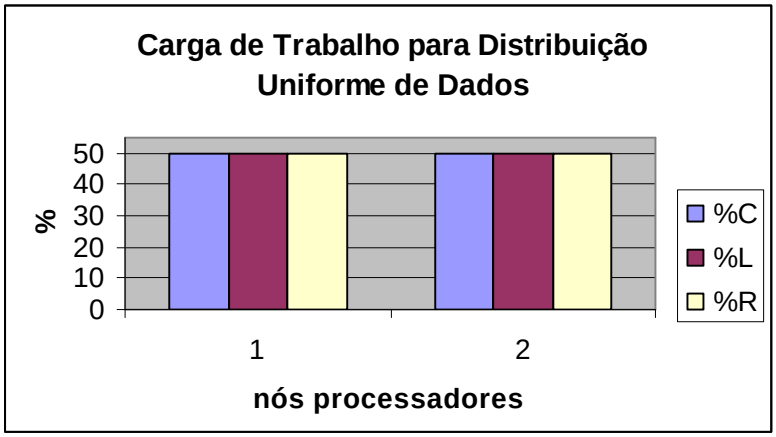

Figura 24: Distribuição ótima para a Junção 1

Exemplo 2: Calcular distribuição ótima da carga de trabalho para Junção 2

Analogamente ao exemplo 1, a partir do Passo 2, obtém-se a tabela representada pela Figura 25. Agrupando-se a carga de trabalho referente a cada processador obtém-se a tabela representada pela Figura 26.

Observe que a diferença de tuplas resultante é igual a 4 (102 - 98). De acordo com o passo 3, deve-se retirar uma tarefa do processador mais carregado que tenha um número de tuplas resultantes iguais a 2, metade do valor da diferença (ou o mais próximo possível e depois tentar equilibrar as tuplas resultantes novamente). No exemplo, existe no processador 1 uma tupla que resulta em 1 resposta. Esta tupla é referente ao valor 18 no atributo de junção. Esta tarefa será transferida para o nó 2. Após executar esta transferência, o número de tuplas lidas de R está desequilibrado pois o processador 2 lê uma tupla de R a mais que o 1.

Segundo o passo 4, se for possível, deve-se equilibrar o número de tuplas de R lidas, transferindo tuplas de R que resultam em zero de S, do nó mais carregado para o nó menos carregado. De acordo com as tarefas distribuídas, pode-se transferir a tarefa referente às tuplas com valor 9 do processador 2 para o 1 e assim obtém-se a distribuição de tarefas representada pela Figura 26.

Valor da Col. C0 de R	Nº de col. C5 de S	Processador que irá executar
1	23	1
2	20	2
6	19	2
15	17	1
14	16	2
10	16	1
11	15	2
7	15	1
12	10	2
8	10	2
3	10	1
16	9	1
4	8	1
17	4	2
13	4	2
19	3	1
18	1	1
5	0	2
9	0	2
20	0	1

Figura 25: Tabela com a distribuição inicial de tarefas da Junção 2

Tuplas de R	Valor da Coluna C0 de R	Nº de col. C5 de S	Processador que irá executar
1	1	23	1
1	15	17	1
1	10	16	1
1	7	15	1
1	3	10	1
1	16	9	1
1	4	8	1
1	19	3	1
1	18	1	1
1	20	0	1
10		102	
1	2	20	2
1	6	19	2
1	14	16	2
1	11	15	2
1	12	10	2
1	8	10	2
1	17	4	2
1	13	4	2
1	5	0	2
1	9	0	2
10		98	

Figura 26: Tabela com estimativa inicial de custo para Junção 2

E assim, a carga de trabalho estará aproximadamente equilibrada entre os processadores 1 e 2 como mostra a tabela e o gráfico da Figura 28.

Tuplas de R	Valor da Coluna C0 de R	Nº de col. C5 de S	Processador que irá executar
1	1	23	1
1	15	17	1
1	10	16	1
1	7	15	1
1	3	10	1
1	16	9	1
1	4	8	1
1	19	3	1
1	9	0	1
1	20	0	1
10		101	
1	2	20	2
1	6	19	2
1	14	16	2
1	11	15	2
1	12	10	2
1	8	10	2
1	17	4	2
1	13	4	2
1	5	0	2
1	18	1	2
10		99	

Figura 27: Tabela com distribuição ótima para Junção 2

Processador	Nº Leitura R	Nº Leitura S	Total Lida	Comparações	Tuplas Result	%C	%L	%R
1	10	101	111	1010	101	50,5	50,45	50,50
2	10	99	109	990	99	49,5	49,55	49,50
			220	2000	200			

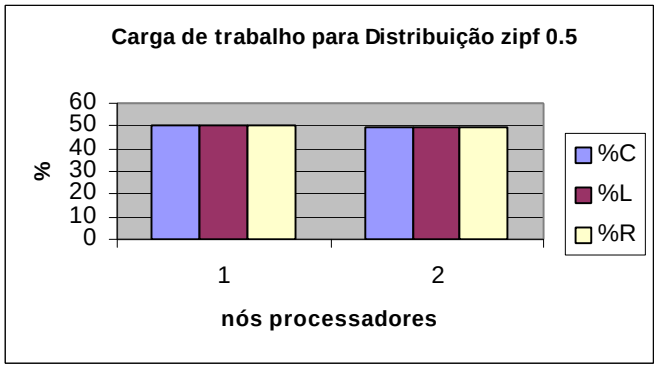

Figura 28: Tabela e gráfico com a distribuição ótima para a Junção 2

Exemplo 3: Calcular distribuição ótima da carga de trabalho para executar a Junção 3

Analogamente aos outros exemplos, o passo 2 da heurística resulta na tabela apresentada pela Figura 29 e o agrupamento inicial das tarefas é apresentado pela Figura 30.

Valor da Col. C0 de R	Nº de col. C10 de S	Processador que irá executar
1	50	1
10	48	2
2	25	2
11	25	1
3	10	2
12	10	1
4	7	2
5	5	1
13	5	2
14	5	2
6	3	1
15	3	1
7	1	1
8	1	2
16	1	2
17	1	1
9	0	1
18	0	2
19	0	2
20	0	1

Figura 29: Tabela com distribuição inicial de tarefas para Junção 3

Observe que a diferença de tuplas resultante neste exemplo também é igual a 4 (102 - 98). De acordo com o passo 3, deve-se retirar uma tarefa do nó mais carregado que tenha um número de tuplas resultantes iguais a 2 (ou o mais próximo possível e depois tentar equilibrar as tuplas resultantes novamente). No exemplo, existe no processador 2 a tupla com valor 8 e a tupla com valor 16 que resultam respectivamente em uma tupla de S. Estas tarefas deverão ser transferidas para o processador 1. Após executar esta transferência, o número de tuplas lidas de R está desequilibrado pois o processador 1 lê duas tuplas a mais que o 2.

Segundo o passo 4, se for possível, deve-se equilibrar o número de tuplas de R lidas, transferindo tuplas de R que resultam em zero de S, do processador mais carregado para o menos carregado. De acordo com as tarefas distribuídas, pode-se transferir as tarefas referentes às tuplas com valores 9 e 20 do nó 1 para o nó 2 e assim obtém-se a distribuição de tarefas representada pela Figura 31.

Tuplas de R	Valor da Coluna C0 de R	Nº de col. C10 de S	Processador que irá executar
1	1	50	1
1	11	25	1
1	12	10	1
1	5	5	1
1	6	3	1
1	15	3	1
1	7	1	1
1	17	1	1
1	9	0	1
1	20	0	1
10		98	
1	10	48	2
1	2	25	2
1	3	10	2
1	4	7	2
1	13	5	2
1	14	5	2
1	8	1	2
1	16	1	2
1	18	0	2
1	19	0	2
10		102	

Figura 30: Tabela com estimativa inicial de custo para Junção 3

Tuplas de R	Valor da Coluna C0 de	Nº de col. C5 de S	Processador que irá executar
1	1	50	1
1	11	25	1
1	12	10	1
1	5	5	1
1	6	3	1
1	15	3	1
1	7	1	1
1	17	1	1
1	8	1	1
1	16	1	1
10		100	
1	10	48	2
1	2	25	2
1	3	10	2
1	4	7	2
1	13	5	2
1	14	5	2
1	9	0	2
1	20	0	2
1	18	0	2
1	19	0	2
10		100	

Figura 31: Tabela com distribuição ótima para Junção 3

Finalmente, a carga de trabalho para este exemplo está igualmente equilibrada entre os processadores 1 e 2 como mostram a tabela e o gráfico da Figura 32.

Processador	Nº Leitura R	Nº Leitura S	Total Lida	Comparações	Tuplas Result	%C	%L	%R
1	10	100	110	1000	100	50	50,00	50,00
2	10	100	110	1000	100	50	50,00	50,00
			220	2000	200			

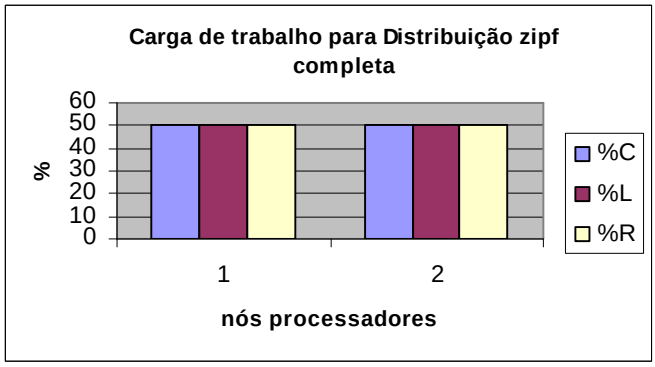

Figura 32: Gráfico com a distribuição ótima da carga de trabalho do Exemplo 3

Exemplo 4: Calcular distribuição ótima da carga de trabalho para a Junção 4

Executando o passo 2 da mesma forma que para os exemplos anteriores, obtém-se os dados representados pela Figura 33.

Valor da Col. C0 de R	Nº de col. C11 de S	Processador que irá
1	120	1
10	10	2
12	10	2
5	9	1
4	7	2
20	6	1
9	5	2
13	5	1
14	5	2
19	4	2
2	3	1
6	3	1
8	3	1
15	3	2
18	3	2
3	1	1
7	1	1
16	1	2
17	1	2
11	0	1

Figura 33: Tabela com distribuição inicial de tarefas para Junção 4

A seguir, as tarefas distribuídas são agrupadas por processador o que possibilita a construção da tabela apresentada na Figura 34.

Tuplas de R	Valor da Coluna C0 de R	Nº de col. C11 de S	Processador que irá executar
1	1	120	1
1	5	9	1
1	20	6	1
1	13	5	1
1	2	3	1
1	6	3	1
1	8	3	1
1	3	1	1
1	7	1	1
1	11	0	1
10		151	
1	10	10	2
1	12	10	2
1	4	7	2
1	9	5	2
1	14	5	2
1	19	4	2
1	15	3	2
1	18	3	2
1	16	1	2
1	17	1	2
10		49	

Figura 34: Tabela com distribuição inicial de tarefas para a Junção Exemplo 4

Observe que a diferença de tuplas resultante é igual a 102 (151-49). De acordo com o passo 3, deve-se retirar uma tarefa do nó menos carregado que tenha um número de tuplas resultantes iguais a 51 (ou o mais próximo possível e depois tentar equilibrar as tuplas resultantes novamente). No exemplo, não há como conseguir esta transferência de tarefas. Sendo assim, a solução indicada pela heurística será transferir o máximo possível de tarefas com tuplas resultantes do nó 1 para o nó 2, o que será caracterizada pela transferência das tarefas referentes aos valores da coluna C0 de R iguais a 5, 20, 13, 2, 6, 8, 3, 7 para o processador 2.

Segundo o passo 4, se for possível, deve-se equilibrar o número de tuplas de R lidas, transferindo tuplas de R que resultam em zero de S, do nó mais carregado para o nó menos carregado. De acordo com as tarefas distribuídas, não existem tarefas com resultado zero que possam ser transferidas do processador 2 para o 1.

Finalmente, executando as transferências de tarefas sugeridas pela heurística, obtém-se a distribuição apresentada pela Figura 35.

Tuplas de R	Valor da Coluna C0 de R	Nº de col. C11 de S	Processador que irá executar
1	1	120	1
1	11	0	1
2		120	
1	10	10	2
1	12	10	2
1	4	7	2
1	9	5	2
1	14	5	2
1	19	4	2
1	15	3	2
1	18	3	2
1	16	1	2
1	5	9	2
1	20	6	2
1	13	5	2
1	2	3	2
1	6	3	2
1	8	3	2
1	3	1	2
1	7	1	2
1	17	1	2
18		80	

Figura 35: Tabela com distribuição ótima da carga de trabalho na Junção 4

Processador	Nº Leitura R	Nº Leitura S	Total Lida	Comparações	Tuplas Result	%C	%L	%R
1	2	120	122	240	120	12	55,45	60,00
2	18	80	98	1440	80	72	44,55	40,00
			220	1680	200			

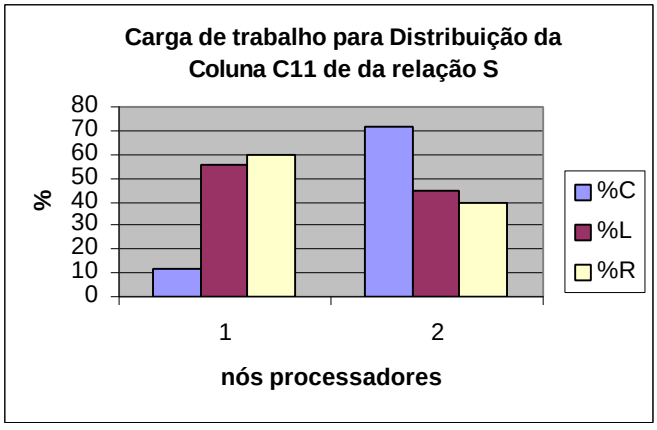

Figura 36: Gráfico com a distribuição ótima para a Junção 4

A carga de trabalho distribuída entre os processadores 1 e 2 é mostrada pela tabela e pelo gráfico da Figura 36. Observe que a divisão de tarefas priorizou a leitura e a escrita de

tuplas que são as operações de maior custo não só na execução da junção como também em SGBDs. Apesar da diferença entre o número de comparação ser aproximadamente 60%, a heurística obteve uma diferença de aproximadamente 10% na taxa de leitura e de 20% na taxa de tuplas resultantes.