UNIVERSIDADE FEDERAL DO ESTADO DO RIO DE JANEIRO
CENTRO DE CIÊNCIAS EXATAS E TECNOLÓGICAS
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA

ANÁLISE E MODELAGEM DE AMBIENTES DE BANCOS DE DADOS DISTRIBUÍDOS

Enrico Francisco Ribeiro de Castro

Orientadora
Morganna Carmem Diniz

RIO DE JANEIRO, RJ – BRASIL
SETEMBRO DE 2010

ANÁLISE E MODELAGEM DE AMBIENTES
DE BANCOS DE DADOS DISTRIBUÍDOS

Enrico Francisco Ribeiro de Castro

DISSERTAÇÃO APRESENTADA COMO REQUISITO PARCIAL PARA OBTENÇÃO DO TÍTULO DE MESTRE PELO PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA DA UNIVERSIDADE FEDERAL DO ESTADO DO RIO DE JANEIRO (UNIRIO). APROVADA PELA COMISSÃO EXAMINADORA ABAIXO ASSINADA.

Aprovada por:

_____
Morganna Carmem Diniz, D.Sc. - UNIRIO

_____
Fernanda Araújo Baião, D.Sc. - UNIRIO

_____
Ana Paula Couto da Silva, D.Sc. - UFJF

RIO DE JANEIRO, RJ – BRASIL
SETEMBRO DE 2010

Castro, Enrico Francisco Ribeiro de,
C355 Análise e modelagem de ambientes de bancos de dados distribuídos / Enrico Francisco Ribeiro de Castro, 2010.
ix, 107f.

Orientador: Morganna Carmem Diniz.
Dissertação (Mestrado em Informática) – Universidade Federal do Estado do Rio de Janeiro, Rio de Janeiro, 2010.

1. Software livre. 2. UAS (UNIRIO analisador de sistemas). 3. Modelagem. 4. Simulação. 5. Banco de dados. I. Diniz, Morganna Carmem. II. Universidade Federal do Estado do Rio de Janeiro (2003-). Centro de Ciências Exatas e Tecnologia. Curso de Mestrado em Informática. III. Título.

CDD 005.3

Agradecimentos

Agradeço à minha orientadora, Profa. Dra. Morganna Carmem Diniz, por acreditar na minha proposta de trabalho, não me deixar abater pelos insucessos momentâneos, e participar ativamente de todas as etapas desta pesquisa. Seu conhecimento e experiência foram fundamentais para conclusão deste trabalho.

Agradeço aos meus pais (Francisco e Maria das Graças) por me incentivarem desde criança a jamais parar de estudar, pois apesar de não ser uma fórmula exata, é a mais próxima para se alcançar o sucesso pessoal e profissional.

Agradeço com muita ênfase à minha esposa Mariana e ao meu filho Bernardo pela imensa compreensão que demonstraram durante estes dois anos e meio nos quais necessitei compartilhar meu tempo entre eles e os estudos. Sem seus incentivos, colaboração e, sobretudo, amor, este sonho não teria sido realizado.

Agradeço à Profa. Dra. Fernanda Baião por colaborar com sugestões preciosas e esclarecimentos de dúvidas durante a realização deste trabalho.

Agradeço aos meus colegas de curso: Ismael Mariano pelas suas excelentes “aulas” de estatística e probabilidade e cadeias de Markov, e, sobretudo por não ter me deixado desistir de tudo no dia em que eu havia tomado esta decisão; Lúcia Castro pelas diversas ajudas prestadas na disciplina e temas relacionados a banco de dados, pelas revisões de texto, e por compartilhar comigo as dificuldades e alegrias desta etapa; Max Faria e Marcelo Castellan pelas dicas técnicas, apoio e incentivo; Marcos Ferreira e Marcos Veloso pelas inúmeras ajudas no laboratório CG-OLAP.

Agradeço ao meu ex-colega de IBM, Bruno Penedo, pelas sugestões e dicas técnicas.

A Deus, por me proporcionar esta experiência e me dar forças para seguir em frente.

CASTRO, Enrico Francisco Ribeiro de. **Análise e Modelagem de Ambientes de Bancos de Dados Distribuídos**. UNIRIO, 2010. 119 páginas. Dissertação de Mestrado. Departamento de Informática Aplicada, UNIRIO.

RESUMO

Este trabalho apresenta o desenvolvimento de um *software* livre denominado UAS (UNIRIO Analisador de Sistemas); esse analisador monitora recursos de ambientes distribuídos, e replica, em um ambiente simulado, o ambiente de rede real monitorado. Isso se dá através da conversão automática dos *logs* do TCPDump em um arquivo de entrada para a ferramenta de simulação de rede NS-2, possibilitando a realização de testes sem que haja necessidade de promover alterações no ambiente real. A solução aqui apresentada constitui um importante artefato para avaliação de desempenho de aplicações com tráfego intensivo de dados em ambientes distribuídos, como ocorre em *clusters* de bancos de dados. A partir da conversão dos *logs* do TCPDump de um ambiente real no arquivo de entrada do NS-2 foram realizados vários testes de simulação de projeção de alteração de configuração de parâmetros de rede.

A outra contribuição deste trabalho é o desenvolvimento de modelos markovianos com apoio da ferramenta TANGRAM-II. Esses modelos representam consultas SQL que utilizam técnicas de fragmentação virtual de dados em um *cluster* de banco de dados. Tais modelos auxiliam na avaliação de desempenho de *clusters* de bancos de dados distribuídos que possuem esta característica. Os ambientes simulado e modelado foram validados a partir da comparação com os ambientes reais. Diversos testes de avaliação de desempenho foram realizados em ambas as soluções.

Palavras-chave: Modelagem, simulação, *cluster*, banco de dados.

ABSTRACT

This research led to the development of a free software named UAS (UNIRIO Analisador de Sistemas), a system analyzer that monitors distributed environment and then replicates it to a simulation one. This simulation is performed by the automatic conversion of the TCPDump logs into an input for NS-2, a network environment simulation tool; this way, the environment can be tested without the need to impose changes to the real network environment. The solution presented in this work is an important artifact for performance evaluation of intensive data traffic applications in distributed environments, like database clusters. From the converted TDPDump logs, several tests were performed using the NS-2 tool, projecting configuration changes in network parameters.

One other contribution presented here is the development of Markovian models with the support of the TANGRAM-II tool. Such models represent SQL queries directed to database clusters that adopt virtual data fragmentation techniques. The models created help evaluate the performance of such distributed database clusters.

Simulated and modeled environments analysis were made by comparing them to the real environments. Several such tests were conducted for both solutions.

Keywords: Modeling, simulation, cluster, database.

Sumário

Lista de Figuras

1 Introdução

A rápida e constante evolução das técnicas de armazenamento, consulta e atualização de dados aliados ao crescimento exponencial do tamanho das bases e à redução gradual dos preços dos computadores pessoais e dispositivos de armazenamento tornam cada vez mais necessário o suporte de alto desempenho para operações em bancos de dados. A utilização de *clusters* de PCs (*Personal Computers*) apresenta-se como uma alternativa de baixo custo para o aumento de desempenho de operações sobre bancos de dados. *Cluster* de banco de dados pode ser definido como um *cluster* de PCs onde cada um destes executa um sistema gerenciador de banco de dados (SGBD) padrão [1].

Clusters de bancos de dados são recomendados quando os dados destas bases são críticos, quando os serviços precisam estar disponíveis, ou ainda quando estes dados necessitam ser processados com maior velocidade. Provedores de acesso à Internet, *sites* de comércio eletrônico, e instituições financeiras costumam utilizar *clusters* de banco de dados como solução para alta disponibilidade, balanceamento de carga e aumento de poder de processamento. Um exemplo do uso deste tipo de infraestrutura é o Google[1], que possui diversos *clusters* com cerca de milhares de máquinas espalhadas ao redor do mundo.

Particularmente, os *clusters* que proporcionam o processamento paralelo aumentam o desempenho de aplicações, sobretudo daquelas que demandam tarefas computacionais de alto custo. Neste tipo de *cluster* as tarefas podem ser divididas em outras menores, distribuídas, e processadas pelos nós participantes. Este recurso cria a ilusão da existência de um supercomputador capaz de processar rapidamente as tarefas

1 http://labs.google.com/papers/googlecluster-ieee.pdf

demandadas pelas aplicações. Geralmente, o *cluster* de processamento paralelo é utilizado para suportar aplicações científicas diversas e operações financeiras, pois estas requerem grande poder de processamento.

Muitas aplicações demandam consultas de alto custo, e a redução do tempo de processamento destas é um dos desafios das pesquisas em bancos de dados. O processamento paralelo através de soluções de *cluster* é uma das alternativas para aumentar o desempenho e diminuir o tempo de resposta das consultas. Em razão do baixo custo, o *cluster* de banco de dados sobre PCs se apresenta como uma solução viável para instituições de todos os portes.

Para tornar possível o paralelismo, cada um dos PCs do *cluster* deve possuir uma réplica inteira ou fragmentos do banco de dados. Tipicamente são utilizadas duas técnicas de paralelismo de consultas em *clusters* de banco de dados: uma onde cada nó do *cluster* processa uma ou mais consultas individuais, e outra onde uma consulta individual é processada simultaneamente por vários nós do *cluster* [2].

Estas duas técnicas são utilizadas em conjunto no ParGRES [2], uma camada que atua como um *software* intermediário (*middleware*) entre a aplicação e o SGBD e coordena as operações dos mesmos com o intuito de executar o processamento paralelo em um *cluster* de PCs. O ParGRES implementa uma técnica de fragmentação de dados com replicação total do banco de dados proposta por LIMA em [3].

A fragmentação de dados é um processo onde as relações são divididas em sub-relações separadas ou fragmentos [4], cujo objetivo é aumentar a eficiência da execução paralela de transações que acessam diferentes sub-conjuntos de tuplas de uma mesma relação. A fragmentação virtual de dados pode ser implementada através de diferentes técnicas. A proposta deste trabalho é discutir e avaliar por meio da simulação e modelagem matemática três destas técnicas: Particionamento Virtual Simples (*Simple Virtual Partitioning* – SVP) [1], técnica que para dividir uma consulta entre os nós de um *cluster* assume distribuição uniforme dos valores associados ao atributo de fragmentação entre as tuplas da relação; Particionamento Virtual Adaptativo (*Adaptive Virtual Partitioning* – AVP) [3], técnica que divide uma consulta entre os nós do *cluster* ajustando dinamicamente os tamanhos dos fragmentos virtuais mesmo não conhecendo nenhuma informação prévia relacionada ao banco de dados e ao SGBD, e; Particionamento Virtual Adaptativo com Redistribuição Dinâmica de Carga **(***Adaptive*

Virtual Partitioning Workload Redistribution – AVP_WR) [3], técnica aperfeiçoada a partir da AVP que para resolver o problema de desbalanceamento de carga, que pode ocorrer nas técnicas SVP e AVP, redistribui dinamicamente a carga de processamento durante a execução de uma consulta.

Embora o processamento paralelo seja uma solução bastante utilizada para reduzir o tempo de resposta de consultas de alto desempenho, não existem estudos específicos baseados na simulação e na modelagem matemática de consultas executadas em bancos de dados que utilizam técnicas de fragmentação virtual de dados. A simulação é a forma mais popular de avaliar sistemas reais, pois pode facilmente ser empregada, uma vez que as operações e carga são descritas através de algoritmos apropriados em programas de computador. Um modelo matemático é um conjunto de equações que se apoiam sobre o sistema real e representam as hipóteses utilizadas na construção do modelo de maneira quantitativa. Estes estudos são importantes para avaliar o desempenho tanto de sistemas já implementados quanto em fase de projeto, ou ainda para prever o comportamento de ambientes antes destes serem submetidos à alterações de configuração. A utilização destes recursos fornece a possibilidade de execução de testes não empíricos nos ambientes e até mesmo avaliá-los antes que sejam implementados.

Um outro problema relacionado aos bancos de dados é a não priorização do planejamento estruturado do ambiente de infraestrutura que os suportam. Cada vez mais explorados, os bancos de dados distribuídos são parte essencial dos negócios de uma instituição. O aperfeiçoamento contínuo destes e de sua infraestrutura faz com que seja cada vez mais necessário o monitoramento, gerenciamento e previsão estruturada de crescimento dos mesmos, a fim de garantir a disponibilidade dos serviços em nível de desempenho estável e aceitável para os usuários.

Apesar da inerente necessidade da existência de um gerenciamento adequado, prevenção de problemas e planejamento de expansão de recursos da infraestrutura que suportam bancos de dados distribuídos, as organizações atuam apenas de forma reativa na identificação e solução de problemas. Normalmente tentam obter um gerenciamento razoável através do investimento em ferramentas de monitoramento do tráfego de rede e desempenho computacional. Em geral promovem o crescimento do ambiente de acordo

com a demanda e sem a realização de um estudo prévio do comportamento futuro do mesmo.

Atualmente existem diversas opções de ferramentas comerciais de monitoramento de recursos de infraestrutura, entretanto, não há muitas opções desenvolvidas em *software* livre. Sobretudo ferramentas que operem com desempenho satisfatório, baixo consumo de recursos computacionais, facilidade de instalação e configuração, e interface simples e amigável. O Cacti [5] apesar de ser bastante flexível e se adaptar com robustez a diversas necessidades, não possui usabilidade simples e intuitiva. Além disso, não possui um utilitário que simplifique a inclusão de um cliente, ou seja, todos os dispositivos precisam ser adicionados ao monitoramento através do preenchimento manual de informações referentes a cada um dos recursos que se deseja monitorar. O NTop [6] possui uma interface amigável através de um *web browser*, porém, todas as parametrizações precisam ser efetuadas através da execução manual de diversos comandos, o que dificulta sua correta configuração. Além disso, é instalado como um serviço, ou seja, existe um processo que roda em *background*, o que ajuda a degradar o desempenho da ferramenta.

Ao contrário de empresas privadas, além dos *softwares* de monitoramento de recursos computacionais, instituições acadêmicas costumam utilizar ferramentas de simulação de redes como o OMNet++ [7] e o NS-2 (*Network Simulator* 2) [8] para estudar novas soluções e configurações antes de implementá-las, o que resulta em economia financeira e redução de imprevistos futuros. Entretanto, a utilização destas ferramentas é uma tarefa que requer conhecimento prévio de linguagens de programação, arquitetura de rede e do ambiente que se deseja avaliar, pois a definição da topologia e dos eventos deve ser realizada em arquivos que são interpretados por estes *softwares*. Infelizmente nem todos os profissionais possuem todas estas habilidades, o que dificulta a utilização e disseminação destas ferramentas, sobretudo nas instituições privadas.

Existe ainda um outro problema relacionado à simulação de ambientes de rede, que é a camada de *software* executada sobre estes. Em simulação é comum abstrair esta camada, contudo, em algumas situações é interessante considerar o tipo de *software* existente em um dado ambiente, pois este determina as características dos dados que trafegam no mesmo. Embora as ferramentas OMNet++ e NS-2 representem com

elevado grau de fidelidade o comportamento do tráfego de rede, elas não possuem variedade de módulos que retratem a camada de *software*. Por isso, muitas vezes é necessário adotar adicionalmente a modelagem matemática na solução de problemas científicos. Abordagem que deve estar integrada e possuir estreito relacionamento com uma linha de pesquisa que necessite de auxílio na solução de problemas complexos. Uma destas é a que estuda a solução de problemas e análise de desempenho de bancos de dados distribuídos.

Ainda que existam diversos modelos matemáticos direcionados à solução de problemas relacionados a banco de dados, não há relatos na literatura da existência de modelos matemáticos aplicados ao estudo do desempenho de *clusters* de bancos de dados distribuídos que utilizam consultas com fragmentação virtual de dados.

Ter o controle total do ambiente é uma questão de sobrevivência para muitas instituições, pois a cada dia que passa novos dados gerados precisam ser analisados e tratados em tempo real. Informações *online* para tomada de decisão devem estar disponíveis em todos os campos da Tecnologia da Informação (TI), pois a melhoria da qualidade de serviços ou prevenção de problemas são quesitos que estão sempre em pauta em todas as organizações. Para analisar o comportamento e o desempenho de um ambiente são necessárias ferramentas e técnicas específicas. Por estes motivos esta pesquisa abordou o desenvolvimento de uma ferramenta de monitoração de ambientes distribuídos, e a simulação e modelagem matemática de ambientes de *clusters* de bancos de dados que utilizam técnicas de fragmentação virtual de dados.

Este trabalho tem os seguintes objetivos:

- Desenvolver uma ferramenta livre de monitoramento do tráfego de rede em ambientes distribuídos. A ferramenta, denominada UAS (UNIRIO Analisador de Sistemas) [9], possibilita o acesso à informações relacionadas à utilização de recursos de computadores em um ambiente distribuído através de informações consolidadas e da geração de gráficos históricos;
- Integrar à UAS um programa que atua como uma interface entre um ambiente de rede monitorado e um ambiente simulado na ferramenta NS-2. Este programa, denominado “TCPDump2NS”, é capaz de coletar informações pertinentes à topologia da rede e seu comportamento e gerar um arquivo de saída que pode ser importado para o NS-2, e desta forma replicar automaticamente o

ambiente real num ambiente simulado, o que possibilita o estudo do comportamento deste ambiente quando submetido a alguma alteração de configuração dos elementos de rede. Ressaltamos que não existe abordagem similar descrita na literatura;

- Desenvolver modelos matemáticos que representam o processamento de consultas SQL (*Structured Query Language*) [10] que utilizam técnicas de fragmentação virtual de dados em bancos de dados distribuídos. Estes modelos possuem características que tornam possível sua aplicação em qualquer sistema de banco de dados distribuído em *cluster*. Devido às propriedades markovianas destes modelos o TANGRAM-II [11] foi a ferramenta de modelagem escolhida para o desenvolvimento dos mesmos.

As propostas apresentadas neste trabalho foram avaliadas comparando os resultados obtidos com os resultados gerados em um ambiente real que faz parte de um projeto financiado pela FAPERJ e desenvolvido pela Universidade Federal do Estado do Rio de Janeiro. O projeto é intitulado “Implementação e Análise de Desempenho de *Clusters* de PCs usando Aplicações OLAP sobre Bancos de Dados Espaciais”, representado pela sigla CG-OLAP, uma abreviação para *Clustered*-GeoOLAP.

Pelo fato deste ambiente possuir uma quantidade limitada de computadores e recursos foram realizadas simulações de situações, tanto com o TCPDump2NS quanto com os modelos desenvolvidos no TANGRAM-II. Nestas simulações foram promovidas mudanças, como por exemplo, a alteração da quantidade de nós do *cluster*, banda de rede disponível, taxa de serviço, entre outros, pois desta maneira é possível identificar problemas e prever o comportamento deste ambiente quando submetido a alguma alteração de configuração.

Ambientes de *clusters* de bancos de dados que utilizam técnicas de fragmentação virtual de dados são complexos. Por isso, é importante estudá-los e avaliá-los com auxílio de recursos de simulação e modelagem matemática. Desta forma, o desenvolvimento de uma ferramenta que facilite a realização de testes de simulação, bem como a criação de modelos markovianos são instrumentos essenciais na tarefa de avaliação desses ambientes.

Este trabalho está organizado conforme a seguir. O capítulo 2 disserta sobre os conceitos de banco de dados e fragmentação virtual de dados, e detalha o ambiente real utilizado na pesquisa. O capítulo 3 descreve o desenvolvimento e as características da ferramenta UAS, sua integração com o TCPDump2NS e os resultados obtidos com o mesmo, enquanto o capítulo 4 discute os modelos matemáticos desenvolvidos e apresenta os resultados alcançados por estes. Finalmente, o capítulo 5 apresenta as conclusões deste trabalho.

2 Fundamentos de Banco de Dados

2.1 Conceitos

Em razão do ambiente de estudo de caso das propostas apresentadas neste trabalho ser um laboratório de pesquisa relacionada a banco de dados geográfico distribuído, este capítulo aborda uma revisão dos conceitos de banco de dados e descreve as técnicas de fragmentação virtual de dados utilizadas neste ambiente.

Banco de dados é um conjunto de dados integrados que atende a um grupo de usuários, e sistema de gerência de banco de dados (SGBD) é um *software* que tem como funções definir, recuperar e alterar dados em um banco de dados [12].

Uma estrutura básica de SGBD geralmente utiliza a arquitetura cliente/servidor. É no módulo cliente que as funcionalidades do sistema são distribuídas, projetado para ser alocado em uma estação de trabalho através de uma interface gráfica amigável com o usuário. O módulo servidor, em geral alocado em computadores de maior porte, é onde os dados são armazenados e onde ocorrem a gestão de acesso, operações de pesquisas e outras funções.

2.2 Modelo de Dados

Um modelo lógico de dados é um conjunto de conceitos que podem ser usados para descrever a estrutura de um banco de dados. É entendida como a estrutura de um banco de dados os tipos de dados, os relacionamentos e as restrições que suportam os dados. Além disso, muitos modelos possuem ainda uma série de operações básicas utilizadas nas tarefas de atualização e recuperação no banco de dados [13].

O modelo lógico de dados mais utilizado atualmente ainda é o modelo relacional, que se caracteriza pelo fato das estruturas possuírem a forma de tabelas, compostas por linhas e colunas.

2.3 Conceitos do Modelo Relacional

O modelo relacional representa o banco de dados como uma coleção de relações, onde cada uma destas se parece com uma tabela de valores. O nome da tabela e das colunas são utilizados para auxiliar na interpretação do que realmente significa os valores contidos em cada linha. Na terminologia do modelo relacional formal, a tabela se chama relação, as linhas são denominadas tuplas, enquanto o cabeçalho das colunas são conhecidos como atributos. Os tipos de valores que podem popular cada uma das colunas são descritos de acordo com os tipos de dados que representam o domínio de valores possíveis.

2.4 Domínios, Atributos, Tuplas e Relações

Domínio é a definição do universo de valores permitidos para um determinado item de dado. Um domínio compreende um tipo, ou formato de dado previamente definido e ainda um grupo de restrições de integridade que tem como objetivo limitar os valores permitidos para este tipo de dado. Desta forma, os valores deste dado se tornam coerentes com a realidade.

Domínios podem ser simples, com um único valor, como um número inteiro por exemplo, ou compostos, com diversos valores, como por exemplo uma data, que possui dia, mês e ano.

Atributo é o nome que se dá a um domínio que representa um dado, ou seja, o nome de um conjunto, uma coluna de uma tabela. Normalmente um atributo apresenta um valor que condiz com o domínio ao qual ele está associado, por exemplo, o atributo CEP é condizente com uma tabela denominada endereço.

Cada linha formada por uma lista ordenada de colunas representa uma tupla, também chamada de registro. As tuplas não precisam obrigatoriamente possuir dados em todas as colunas, caso seja necessário podem assumir também valores nulos. Uma tupla é uma instância de uma tabela.

Relação ou entidade, no modelo relacional, é uma tabela formada por um cabeçalho, que é o conjunto de atributos que deve possuir nomes distintos com intuito de evitar a ambiguidade na localização de uma informação, e um corpo, que é o conjunto de quantidades variáveis de tuplas.

2.5 Cluster de Banco de Dados

Um *cluster* pode ser definido como um sistema computacional local que abrange um conjunto de computadores independentes interconectados por uma rede [14].

Com a redução gradual dos preços dos computadores pessoais, a utilização da tecnologia de *clusters* de PCs (*Personal Computers*) em ambientes de banco de dados distribuídos tornou-se uma solução viável para atender a demanda por poder computacional em aplicações científicas e comerciais pelo fato de possuir uma atrativa relação custo-benefício e ainda ser capaz de fornecer alto poder computacional, confiabilidade e flexibilidade.

A comunicação entre as aplicações localizadas nos *clusters* é realizada através da implementação de bibliotecas de troca de mensagens entre os membros, que geralmente se valem dos protocolos TCP/IP como meio de transporte. Este processo é de vital importância e indispensável quando aplicações são executadas paralelamente. Entretanto, apesar de oferecer inúmeros benefícios, o *cluster* também pode apresentar problemas, como por exemplo a velocidade da comunicação através da rede, que pode ser considerada um gargalo do processo quando comparado à velocidade dos processadores, que geralmente, no que tange à velocidade, avançam mais rapidamente que a banda de rede. Por este motivo faz-se necessário uma análise criteriosa de um ambiente que usa esta tecnologia, pois muitas são as variáveis que podem causar influência sobre o desempenho.

Um *cluster* de bancos de dados pode ser alocado em PCs, bastando para isso executar em cada nó um sistema gerenciador de bancos de dados padrão. Os *clusters* de bancos de dados são sistemas caracterizados por possuir arquitetura do tipo memória distribuída. Esta arquitetura tem a expansibilidade como característica principal, o que facilita a expansão do sistema através da adição de unidades de processamento. Este recurso é bastante dificultado quando as arquiteturas de memória e disco são compartilhadas.

2.6 Banco de Dados Distribuídos

Um banco de dados distribuído (BDD) é uma coleção de múltiplos bancos de dados inter-relacionados logicamente e distribuídos através de uma rede de computadores. Um sistema de gerenciamento de banco de dados distribuídos (SGBDD) é o sistema de *software* que gerencia o BDD e torna transparente a distribuição para o usuário [13].

O banco de dados distribuído surgiu a partir da integração entre tecnologia de um banco de dados centralizado com as redes de computadores distribuídas. Bancos de dados distribuídos devem possuir independência dos dados, transparência de rede ou localização, de replicação e de fragmentação.

A transparência de rede ou localização fornece ao usuário a ilusão que o dado é local. A transparência de replicação é mantida quando mesmo ao se possuir replicações espalhadas pelo sistema, ou seja, pelos dispositivos que armazenam uma réplica deste banco de dados, estas sejam todas consistentes. E a transparência de fragmentação mantém a consistência e a integridade em cada fragmento de dados.

2.7 Banco de Dados Geográficos

Sistemas de Informação Geográfica (SIGs) são sistemas que tratam dados espaciais computacionalmente por intermédio das seguintes funcionalidades: entrada e validação; armazenamento e gerenciamento; saída e apresentação visual; transformação; interação com o usuário; e combinação para criar novas representações do espaço geográfico [15].

Um banco de dados geográfico tem como objetivo o fornecimento de meios que tornem possível a manipulação de dados georreferenciados com a mesma simplicidade com a qual se manipula dados em um banco de dados relacional.

Uma vez que estes dados existam, essas informações devem ser obtidas com facilidade, e para que isso seja possível foi desenvolvida uma linguagem de manipulação parecida com o SQL, que na realidade atualmente é uma extensão do próprio SQL. Por esta razão, os formatos de dados básicos não são suficientes, e sendo assim, novos tipos de dados, funções e operadores foram criados para tornar possível a obtenção da mesma capacidade de manipulação que existe em um banco de dados

relacional. São necessários diversos tipos geométricos de dados, como linha, ponto, círculo, polígono, etc, e também operações sobre estes, como distância, área, adjacência, etc.

2.8 Fragmentação de Dados

Fragmentação de dados é o processo de divisão de relações em sub-relações separadas ou fragmentos [4], e tem como objetivo aumentar a eficiência da execução paralela de transações que acessam diferentes sub-conjuntos de tuplas de uma mesma relação. A fragmentação de dados pode ser física ou virtual.

Na fragmentação física, ou real, os dados são divididos ao longo de todo o sistema, ou seja, cada nó possui uma base de dados diferente se a considerarmos sob o aspecto local, porém, se analisarmos maneira global, os dados são vistos de forma única, pois cada nó do *cluster* possui um catálogo que contém todas as informações dos dados dos bancos adjacentes.

Para se implementar a fragmentação virtual é necessário que cada nó do *cluster* seja capaz de acessar todo o banco de dados, o que pode ser realizado caso a base de dados seja totalmente replicada em uma arquitetura de disco ou memória distribuída. Quando o *cluster* recebe uma consulta ele a reescreve como uma sequência de sub-consultas, adiciona os predicados responsáveis por definir os intervalos que correspondem aos diferentes fragmentos virtuais de uma relação, e envia uma para cada nó. Desta maneira as sub-consultas podem ser processadas paralelamente por todos os nós. A fragmentação virtual é vantajosa em relação à física nos seguintes aspectos: não necessita de um projeto de BDD, e oferece flexibilidade para que os fragmentos sejam definidos e alocados nos nós no momento do processamento de consultas [3].

A fragmentação virtual é realizada de maneira oposta aos fragmentos reais criados pelas técnicas que executam a fragmentação física da base de dados. Suponha, por exemplo, que uma consulta **C** que necessite de dados da tabela **T** seja submetida a um *cluster* de banco de dados. O processador de consultas, que neste trabalho será referido como “nó mestre” escolherá um grupo de nós, referidos como “nós escravos”, para processar **C**. Cada um destes nós processará então um ou mais fragmentos virtuais (FV) diferentes da tabela **T**. Para que isso aconteça *n* sub-consultas ($\mathbf{C}_1$, $\mathbf{C}_2$, ..., $\mathbf{C}_n$) devem ser produzidas, onde cada sub-consulta $\mathbf{C}_i$ é obtida através da substituição de **T**

em **C** por um fragmento virtual $\mathbf{T}_{FVi}$ diferente. Finalmente cada sub-consulta é enviada para um nó diferente.

A fragmentação virtual pode ser implementada como particionamento virtual simples [1] ou como particionamento virtual adaptativo [3]. A seguir, estas duas técnicas são discutidas com mais detalhes.

2.9 Particionamento Virtual Simples (*Simple Virtual Partitioning* – SVP)

Particionamento Virtual Simples é uma técnica que para dividir uma consulta entre os nós de um *cluster* assume distribuição uniforme dos valores associados ao atributo de fragmentação entre as tuplas da relação [1]. Obtém-se os tamanhos dos intervalos que correspondem aos fragmentos por meio do cálculo da diferença entre o maior e o menor valores do atributo de fragmentação contidos nas tuplas. Posteriormente essa diferença é dividida pelo número de nós do *cluster* usados no processamento da consulta. Em seguida os fragmentos são então definidos por intervalos do mesmo tamanho. Desta forma, ao processar uma consulta, o SVP utiliza um número de fragmentos virtuais equivalente ao número de nós do *cluster*. Consequentemente, se *n* nós são usados, *n* sub-consultas são geradas e, a cada nó, é atribuída apenas uma delas.

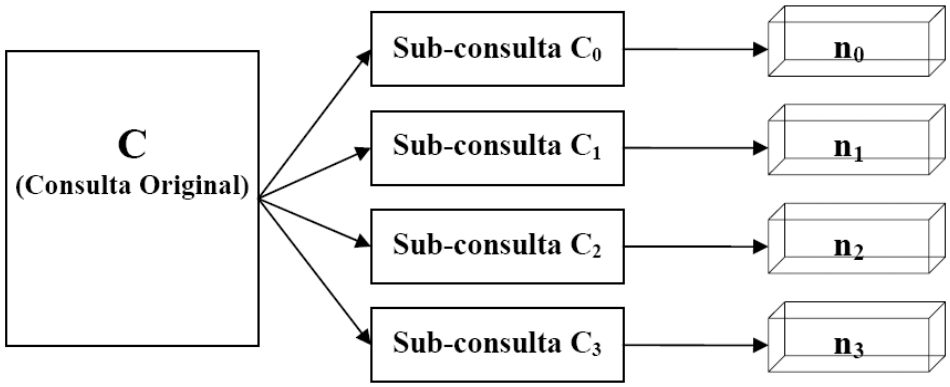

Figura 1: Consulta com emprego da técnica SVP em um *cluster* com 4 nós.

A figura 1, extraída de [3], mostra a metodologia de divisão das sub-consultas realizadas pelo SVP em um *cluster* de 4 nós.

Apesar de funcionar muito bem, esta técnica apresenta dois problemas:

(i) Grandes fragmentos virtuais - durante o processamento das sub-consultas o tamanho do fragmento virtual que será usado é determinado pela cardinalidade da

relação virtualmente fragmentada e pela quantidade de nós utilizados no *cluster*. Quando há um número fixo de nós, quanto maior a cardinalidade da relação, maiores serão as dos seus fragmentos. Em vários SGBDs, quando o número estimado de tuplas de uma relação a ser acessada atinge determinado limite durante o processamento de uma consulta, essa relação é acessada através do uso da busca linear. Portanto, se um ou mais nós precisarem realizar busca linear, o ganho de desempenho obtido com o uso de paralelismo será prejudicado, e o o único meio de evitar este problema é aumentando o número de nós do *cluster*;

(ii) Distorção de dados - acontece quando os valores de alguns atributos não se encontram uniformemente distribuídos entre as tuplas de uma relação [16]. A distorção de valores de atributos pode fazer com que o SVP produza fragmentos virtuais de tamanhos muito diferentes quando comparados à quantidade de tuplas. Mesmo que o tamanho dos intervalos sejam iguais, uma quantidade diferente de tuplas pode estar presente em cada um deles, o que gera o desbalanceamento inicial de carga entre os nós participantes. Desta maneira, quanto maior for a distorção, mais severo será o desbalanceamento.

2.10 Exemplo de uma Consulta com Emprego da Técnica SVP

Para execução do particionamento virtual de consultas é necessário que alguma das colunas de uma tabela a ser consultada esteja "*clusterizada*", ou seja, indexada como chave da busca para um *cluster* SVP.

Suponha que em um banco de dados exista uma tabela de nome "endereço", e que esta possua uma coluna "*clusterizada*" denominada "CEP" que admita valores do tipo inteiro entre "00000000" e "99999999", porém, com dados em apenas 12 tuplas. A saber:

- 22000000, 22000001, 22000002, 23000000, 23000001, 24000000, 24000001, 25000000, 25000001, 26000000, 27000000 e 30000000.

Agora suponha que este banco de dados, replicado em um *cluster* de 4 nós que utiliza a técnica SVP, receba de uma aplicação qualquer uma consulta SQL com a sintaxe "*select * from endereço*", que tem como objetivo verificar todos os endereços cadastrados neste banco. Neste momento o nó mestre calculará a diferença entre o maior

e menor valores da coluna “CEP”, ou seja, “30.000.000 – 22.000.000”, o que resulta no valor “80.000.000”. Em seguida o valor obtido será dividido pela quantidade de nós escravos existentes no *cluster*, neste caso 4, o que resulta no valor “20.000.000”. Posteriormente o SVP adiciona predicados a esta consulta de forma que cada nó escravo seja responsável pelo processamento de uma determinada parcela do processamento, ou seja, uma sub-consulta.

Apesar da tentativa do SVP em realizar a distribuição de carga de forma equânime para todos os nós participantes, nem sempre isso possível. Este exemplo é uma situação onde haverá desbalanceamento de carga, pois cada nó será responsável por processar quantidades diferentes de dados, conforme as sub-consultas descritas a seguir:

- O nó 1 receberá a sub-consulta “*select * from endereço where CEP >= 22.000.000 and <= 23.999.999*”. Desta forma, este nó processará os dados da tabela “endereço” onde o valor das tuplas na coluna “CEP” for “22.000.000, 22.000.001, 22.000.002, 23.000.000 e 23.000.001”, pois no intervalo definido para este nó apenas estas cinco linhas realmente possuem informações, ou seja, o nó 1 processará dados indexados por cinco tuplas.
- O nó 2 receberá a sub-consulta “*select * from endereço where CEP >= 24.000.000 and <= 25.999.999*”. Desta forma, este nó processará os dados da tabela “endereço” onde o valor das tuplas na coluna “CEP” for “24.000.000, 24.000.001, 25.000.000, e 25.000.001”, pois no intervalo definido para este nó apenas estas quatro linhas realmente possuem informações, ou seja, o nó 2 processará dados indexados por quatro tuplas.
- O nó 3 receberá a sub-consulta “*select * from endereço where CEP >= 26.000.000 and <= 27.999.999*”. Desta forma, este nó processará os dados da tabela “endereço” onde o valor das tuplas na coluna “CEP” for “26.000.000 e 27.000.000”, pois no intervalo definido para este nó apenas estas duas linhas realmente possuem informações, ou seja, o nó 3 processará dados indexados por duas tuplas.
- O nó 4 receberá a sub-consulta “*select * from endereço where CEP >= 28.000.000 and <= 30.000.000*”. Desta forma, este nó processará os dados da tabela “endereço” onde o valor das tuplas na coluna “CEP” for “30.000.000”,

pois no intervalo definido para este nó apenas esta linha realmente possui informações, ou seja, o nó 4 processará dados indexados por apenas uma tupla.

Apesar do tamanho definido para cada intervalo ser igual, existe um número diferente de tuplas em cada um deles, o que acarreta no desbalanceamento de carga entre os nós deste *cluster*. Portanto, para diminuir a possibilidade de desbalanceamento carga o ideal é que o número de tuplas existentes em cada intervalo seja sempre aproximadamente o mesmo. Entretanto, como o SVP submete apenas uma sub-consulta a cada nó, e o SGBD não é sempre o mesmo em cada banco de dados, o que dificulta a criação de um algoritmo genérico para solução deste problema, não é trivial adotar uma técnica que realize a redistribuição dinâmica de carga a partir do momento em que cada nó inicia o processamento de sua sub-consulta.

Para solucionar este problema foi desenvolvida por [3] uma técnica denominada AVP (*Adaptive Virtual Partitioning*), ou Particionamento Virtual Adaptativo, que consiste na redistribuição dinâmica de carga entre os nós do *cluster* durante o processamento de uma consulta. Esta técnica está descrita na próxima seção.

2.11 Particionamento Virtual Adaptativo (*Adaptive Virtual Partitioning* - AVP)

Nesta solução o autor propôs uma solução de fragmentação virtual de consultas em um *cluster* de banco de dados que usa técnica de paralelismo intra-consulta (intra-c), que é utilizada quando se deseja reduzir o tempo de processamento de consultas individuais de alto custo. Foi desenvolvido um mecanismo que ajusta dinamicamente os tamanhos dos fragmentos virtuais mesmo não conhecendo nenhuma informação prévia relacionada ao banco de dados e ao SGBD, o que resolve os problemas ligados a grandes fragmentos virtuais [3].

Esta abordagem para a fragmentação virtual não utiliza a mesma estratégia da SVP, onde é processada uma sub-consulta em cada nó existente no *cluster*, mas sim a execução de uma fragmentação virtual de granularidade mais fina, ou seja, usa pequenos fragmentos em quantidade tipicamente maior do que o número de nós utilizados para o processamento desta consulta.

Cada um dos nós do *cluster* recebe uma sub-consulta e um intervalo que corresponde a um fragmento virtual, que tem seu tamanho calculado da mesma forma que no SVP. Os valores que correspondem ao início e final do fragmento virtual da sub-consulta são parâmetros. A mesma sub-consulta é enviada para todos os nós, entretanto, cada um recebe um determinado intervalo na forma de parâmetros. Após esta tarefa o fragmento virtual é processado de forma distinta da proposta pelo SVP [3].

A figura 2, extraída de [3], exibe a metodologia de divisão das sub-consultas realizadas pelo AVP em um *cluster* de 4 nós.

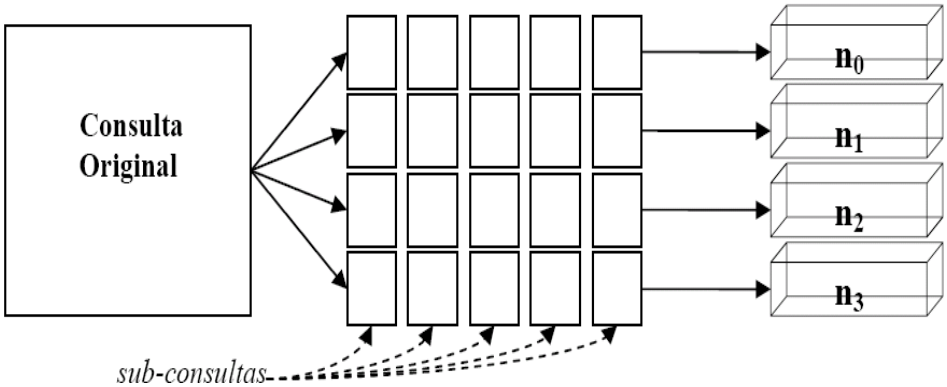

Figura 2: Consulta com emprego da técnica AVP em um *cluster* com 4 nós.

Dividindo, em cada nó, o intervalo inicial em partes menores, se encerra a relação entre o tamanho do fragmento virtual e o número de nós do *cluster* [3]. Desta forma os sub-intervalos podem ser pequenos o bastante para evitar o processamento por meio de buscas lineares executadas pelo SGBD sobre a relação fragmentada.

2.12 Particionamento Virtual Adaptativo com Redistribuição Dinâmica de Carga (*Adaptive Virtual Partitioning Workload Redistribution* - AVP_WR)

O AVP_WR (*Adaptive Virtual Partitioning Workload Redistribution*) é um aperfeiçoamento do AVP. Ele foi desenvolvido por [3] com o objetivo de reduzir ainda mais a possibilidade de haver desbalanceamento de carga durante uma consulta. Este problema pode ocorrer caso os fragmentos iniciais tenham tamanhos desiguais em relação à quantidade de tuplas, pois nesta situação alguns nós poderiam receber uma carga de trabalho maior que outros. Para resolver este problema foi criada uma estratégia de redistribuição dinâmica de carga durante a execução de uma consulta.

No AVP um nó processa os fragmentos virtuais de maneira progressiva por meio de uma sequência de sub-consultas, onde cada uma corresponde a uma pequena parcela do fragmento inicial. Esta característica permite que seja realizada uma interferência e modificação da carga de trabalho de um nó antes mesmo que ele termine de processar seu fragmento inicial [3].

A estratégia adotada para execução da redistribuição dinâmica pelo AVP_WR é totalmente descentralizada e executada de forma simultânea por todos os nós envolvidos no processamento da consulta. Caso a distribuição de carga inicial esteja desbalanceada, o nó que terminar de processar primeiro sua parcela oferece ajuda aos demais, e então, caso algum nó aceite a oferta, uma parcela de carga inicialmente destinada a este nó é então repassada ao nó ofertante.

O AVP_WR se baseia em uma organização lógica dos nós e em um mecanismo de difusão de mensagens. A metodologia de propagação de mensagens é detalhadamente descrita em [3].

O AVP e o AVP_WR foram implementados como parte do *software middleware* ParGRES [2], utilizado no projeto CG-OLAP, ambiente de estudo de caso utilizado nesta pesquisa. O ambiente CG-OLAP e o ParGRES são descritos nas próximas seções deste capítulo.

2.13 Ambiente CG-OLAP

Devido ao crescente volume de dados tratados e à complexidade das consultas feitas à base de dados, o desempenho dos sistemas de informação se tornou um fator crítico nas organizações [17]. Diversas soluções que objetivam aumentar o desempenho entre as operações de consulta, resposta e atualização em banco de dados podem ser adotadas, entre as quais o *cluster*, que foi a escolhida pelo projeto desenvolvido pela Universidade Federal do Estado do Rio de Janeiro, intitulado de “Implementação e Análise de Desempenho de *Clusters* de PCs usando Aplicações OLAP[2] sobre Bancos de Dados Espaciais”, e é representado pela sigla CG-OLAP, uma abreviação para *Clustered*-GeoOLAP.

2 On-line Analytical Processing (OLAP) é uma tecnologia de aplicação capaz de manipular e analisar um grande volume de dados sob múltiplas perspectivas.

O projeto CG-OLAP está dividido em 3 frentes macros de pesquisa: aplicação, banco de dados distribuído e análise de desempenho através da simulação e modelagem matemática. Todas visam investigar e recomendar metodologias e ferramentas computacionais que contribuam para incrementar o desempenho do processamento de um grande volume de dados em um sistema de banco de dados executado sobre um *cluster* de PCs, que pelo fato de possuírem baixo custo são acessíveis até mesmo à pequenas instituições.

2.14 Arquitetura do Projeto CG-OLAP

A infraestrutura definida para o projeto CG-OLAP é composta de seis PCs: quatro nós escravos, responsáveis pelo armazenamento do banco de dados replicado; um nó mestre, que além de também possuir uma réplica do banco de dados é responsável pela divisão do processamento de consultas e atualização de dados, e união e envio das respostas para originador de requisições, e; um nó responsável por originar as tarefas de consulta e atualização de dados.

A configuração básica de cada um dos cinco nós que formam o *cluster* CG-OLAP e do equipamento de rede consiste de: processador Pentium Core 2 Duo, 2 GB de RAM, 2 discos de 250 GB, duas placas de rede Ethernet 10/100/1000, um *switch* camada 2, e cabeamento par trançado CAT5E de sete metros interligando todos os seis PCs com o *switch*. A única informação relevante referente ao PC responsável pela requisição ao *cluster* CG-OLAP é a placa de rede, que a exemplo das demais também é Ethernet 10/100/1000. As demais configurações serão omitidas neste trabalho porque os resultados do estudo não são influenciados por estas.

A figura 3 descreve a arquitetura física do *cluster* CG-OLAP. A máquina denominada “originador de requisições” pode ser utilizada como geradora de consultas para o *cluster*, entretanto, sua utilização não é obrigatória na solução, pois as consultas podem ser executadas a partir do nó processador de consultas, denominado CQP. Todas as consultas realizadas neste trabalho foram executadas a partir deste nó.

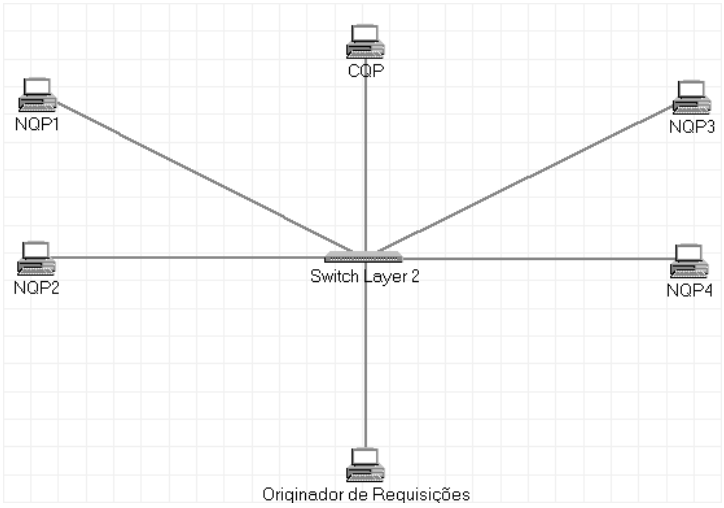

Figura 3: Arquitetura física do *cluster* CG-OLAP

A arquitetura lógica é baseada na arquitetura proposta no Projeto ParGRES [2], um *software* que tem como objetivo possibilitar o processamento paralelo de consultas em um *cluster* de banco de dados. Atua como uma camada intermediária (*middleware*) entre a aplicação e o SGBD e coordena as operações dos mesmos com o intuito de executar o processamento paralelo em um *cluster* de PCs. A característica de paralelismo do ParGRES é destinada à consultas que demandam grande carga de trabalho. Ele concilia diversas técnicas de processamento paralelo de consultas em *clusters* de banco de dados e trata tarefas de atualização de dados e balanceamento de carga entre os nós. A comunicação do ParGRES com o SGBD se baseia no padrão SQL, portanto, não dependente das características de um SGBD específico [18].

No ParGRES existem dois tipos componentes: globais e locais. Componentes globais são aqueles que executam tarefas que envolvem diversos nós do *cluster*, enquanto os locais executam tarefas em apenas um nó. Componentes globais são o "Intermediador" e o "Processador de Consultas de *Cluster"* (CQP – *Cluster Query Processor*). Componentes locais são o "Processador de Consultas de Nó" (NQP – *Node Query Processor*) e o SGBD PostgreSQL [19]. O componente mais importante e que atua como coordenador de todos os demais é o CQP. Ele é o responsável pelo controle da execução de requisições no ParGRES [2]. Neste trabalho o CQP é referido como "nó mestre" e os NQPs como "nós escravos".

A figura 4, extraída de [2] descreve o fluxo de uma consulta enviada ao banco de dados localizado nesta infraestrutura. Caso seja utilizada alguma aplicação cliente, esta

pode estar localizada no PC denominado “originador de requisições”, descrito na figura 3.

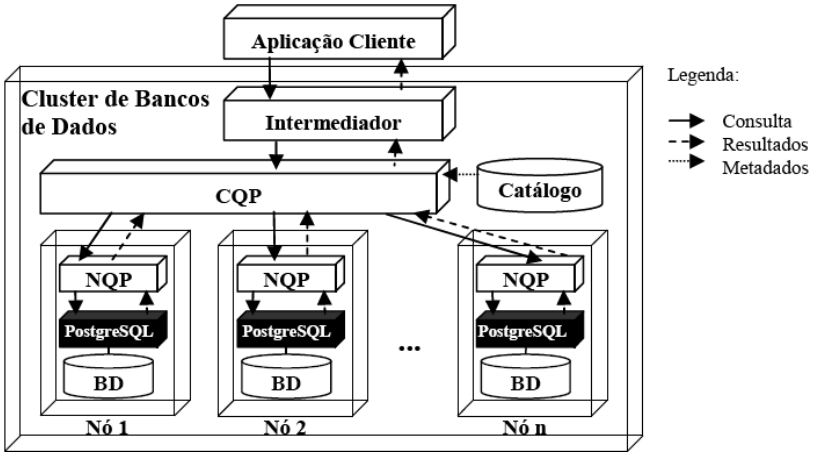

Figura 4. Arquitetura lógica do *cluster* CG-OLAP

O fluxo normal de uma tarefa de consulta no *cluster* CG-OLAP tem início no nó denominado “Originador de Requisições”. Ele envia uma consulta SQL para o nó CQP, e este por sua vez distribui a consulta de forma balanceada entre os nós escravos, denominados NQPs, que processam suas porções de consulta e entregam o resultado ao CQP, que executa a união de todas as porções e finalmente devolve o resultado ao “Originador de Requisições”. Na próxima seção a execução das consultas serão descritas de forma mais detalhada.

2.15 Tarefas Executadas pelo ParGRES

O ParGRES executa quatro tipos de tarefas: (i) tradução da consulta SQL para permitir a execução paralela, (ii) processamento de consultas com paralelismo inter/intra-consultas, (iii) composição de resultados e (iv) processamento de atualização.

O CQP analisa as consultas recebidas da aplicação cliente para então decidir qual tipo de paralelismo será empregado no processamento e quais nós NQPs serão utilizados em cada uma delas. Para realizar estas escolhas o CQP se vale das informações contidas no seu catálogo.

O paralelismo intra-consulta (intra-c) decompõe consultas complexas em sub-consultas que serão executadas paralelamente. Cada sub-consulta em um fragmento de dados diferente. Dessa forma, uma vez que todos possuem replicação total de dados, cada sub-consulta poderá ser enviada para cada um dos nós do *cluster*. Assim, cada sub-

consulta é enviada para um nó diferente e executada em paralelo com as demais. Nesta técnica o CQP aloca os NQPs e envia a cada um deles um plano de execução local para a consulta. Cada NQP processa o seu plano e gera um resultado parcial, que é enviado ao CQP para a composição do resultado final da consulta. Após receber os resultados parciais de todos os NQPs, o CQP finaliza a composição de resultados e os envia à aplicação cliente [18].

No paralelismo inter-consultas (inter-c), ou seja, quando se deseja obter maior vazão em um banco de dados, consultas distintas são executadas de forma concorrente no *cluster* de banco de dados, uma em cada nó do *cluster* [18]. Neste tipo de paralelismo o CQP recebe a consulta da aplicação cliente e a envia apenas para o nó NQP que estiver com a menor quantidade de tarefas pendentes no momento, e este a repassa ao SGBD, que a processa e devolve o resultado ao NQP, que o encaminha para CQP, que finalmente o entrega para a aplicação que originou a requisição.

Como várias consultas podem ser processadas simultaneamente no *cluster*, inclusive pelos mesmos nós, as duas técnicas de paralelismo podem ser utilizadas ao mesmo tempo, pois algumas consultas não requerem alto processamento, e desta forma não se justifica o emprego da técnica intra-c, sendo então a inter-c a mais apropriada.

Os NQPs, através de trocas de mensagens entre si, são os responsáveis pela implementação de uma técnica de balanceamento dinâmico que tem por objetivo equilibrar a carga dos nós envolvidos no processamento de uma mesma consulta. Esta técnica foi proposta em [3].

2.16 Processo Global de Execução de uma Consulta no ParGRES

Suponha, por exemplo, que o nó CQP tenha submetido uma consulta aos nós NQPs. Os NQPs então recebem a sub-consulta parametrizada e um intervalo para ser processado através dela, o que corresponde a um fragmento virtual. O intervalo inicial enviado para cada nó é definido como na técnica SVP, ou seja, calculando-se a diferença entre o maior e o menor valores do atributo de fragmentação presentes nas tuplas e dividindo-se essa diferença pelo número de nós utilizados para o processamento da consulta.

Suponha então que um nó NQP_i receba um intervalo $\mathbf{I}_i$ e uma sub-consulta parametrizada $\mathbf{C}_i$ para processar o seu fragmento virtual. No lugar de processar

inteiramente $\mathbf{I}_i$ através de uma única submissão da sub-consulta, o nó NQP_i subdivide esse intervalo em *p* intervalos menores ($\mathbf{I}_{ip-1}$, ... , $\mathbf{I}_{i2}$, $\mathbf{I}_{i1}$, $\mathbf{I}_{i0}$). O valor deste intervalo *p* deve ser tal que os sub-intervalos $\mathbf{I}_i$ sejam pequenos o bastante. A sub-consulta $\mathbf{C}_i$ é então submetida sequencialmente ao SGBD tantas vezes quanto for o total de sub-intervalos *p*. Em cada uma delas seus parâmetros são substituídos de forma a fazer com que cada sub-intervalo $\mathbf{I}_i$ seja processado. A partir deste momento os nós executam uma série de tarefas para adaptação dinâmica do tamanho dos fragmentos. A figura 5, extraída de [3], ilustra este processo, ou seja, a estratégia da divisão do intervalo inicial de uma consulta no nó NQP_i em *n* sub-intervalos antes de ser entregue ao SGBD.

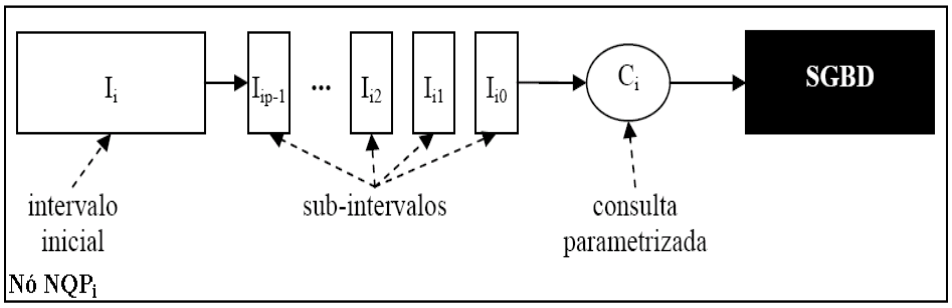

Figura 5: Fragmentação virtual com granularidade fina em um nó NQP_i.

2.16.1 Fases de Consulta

O CQP elege como coordenador do seu processamento o nó que estiver executando menos tarefas no momento. O coordenador é o responsável pela distribuição de carga inicial entre os nós envolvidos no processo, denominados nós participantes. O processamento da consulta se dá então em quatro fases macros:

1) O nó coordenador envia para todos os nós NQPs fragmentos virtuais com intervalos de tamanhos idênticos.

2) Os nós NQPs executam tarefas para adaptação dinâmica do tamanho dos fragmentos.

3) Os nós NQPs utilizam uma técnica para realizarem entre si a redistribuição dinâmica de carga. Esta fase é executada de forma paralela à fase 2.

4) O nó coordenador finaliza a execução paralela da consulta e devolve o resultado ao CQP.

2.16.2 Processamento da Sub-consulta

O CQP, quando submete uma consulta aos nós NQPs, elege o nó coordenador, que pode ser ele próprio, caso esteja executando menos tarefas no momento, e o coordenador então cria uma tarefa local denominada GRC (*Global Result Collector*), que é responsável pela coleta de todos os resultados parciais enviados pelos NQPs, pois as sub-consultas executadas produzem resultados parciais da consulta submetida pelo CQP. Desta maneira a produção do resultado final exige um trabalho de composição global desses resultados parciais, e este trabalho é executado pelo GRC, que ao finalizá-lo envia o resultado definitivo ao CQP.

É importante ressaltar que o nó CQP (nó mestre) poderá desempenhar simultaneamente 3 tarefas globais distintas em uma mesma consulta: (i) CQP, componente que recebe a consulta de uma aplicação e a submete aos nós NQPs; (ii) Coordenador, componente alocado em algum dos nós que é responsável pela realização da fragmentação virtual inicial entre os nós NQPs, ou seja, efetua a distribuição de carga inicial entre os nós envolvidos no processo, e; (iii) GRC, componente sempre alocado junto ao nó coordenador, que é responsável por receber todos os resultados parciais, compô-los, e submeter então o resultado final ao componente CQP.

2.16.3 Redistribuição Dinâmica de Carga

Durante toda o processo de execução de uma consulta os nós NQPs se comunicam com o objetivo de oferecer ajuda a outro nó que ainda não tenha finalizado sua tarefa.

Após receber seu fragmento virtual o nó participante inicia o processamento do mesmo utilizando a AVP. Quando este fragmento estiver totalmente processado o nó informa o coordenador e se torna um “nó livre”, continuando, contudo, no conjunto de nós participantes. Em seguida este nó executa o algoritmo de redistribuição dinâmica de carga, que se baseia em “ofertas de ajuda”, ou seja, ele envia uma mensagem de oferta de ajuda para os demais nós. Caso exista algum nó necessitando de ajuda este envia ao ofertante uma mensagem de aceitação. É importante ressaltar que o nó que aceitou a ajuda, e que portanto ainda está ocupado, não encerra o processamento de sub-consultas enquanto espera a resposta do nó ofertante, ou seja, ele envia a mensagem de aceitação e continua processando seus fragmentos virtuais.

Quando o nó ofertante recebe a primeira mensagem de aceitação interrompe a espera e requisita ao nó que aceitou a oferta o envio de parte do intervalo do fragmento virtual que ainda não foi processado. Quando recebe os fragmentos virtuais, o nó ofertante inicia a execução da sub-consulta parametrizada e o novo intervalo utilizando a AVP. Caso receba outras mensagens de aceitação durante este período, estas são descartadas.

2.16.4 Encerramento da Consulta

O encerramento do processamento de uma consulta é de responsabilidade do coordenador, que necessita saber se todos os fragmentos virtuais foram processados. Para isso foi definido o conceito de nó "proprietário" de um fragmento virtual, onde cada um é proprietário do fragmento recebido do coordenador. Cada nó é responsável por notificar o coordenador quando terminar de processar o seu fragmento, ou seja, cada nó executa esta tarefa cada vez que conclui o processamento de um fragmento virtual sob sua responsabilidade.

2.17 Consultas Efetuadas no *Cluster* CG-OLAP

Os dados utilizados como base para criação do modelo computacional, descritos no capítulo 4, foram coletados a partir da execução de consultas no *cluster* CG-OLAP. Estas consultas foram baseadas no *Star Schema Benchmark* criado por O' NEIL *et al* [20] para possibilitar a comparação de como a paralelização das consultas, valendo-se das técnicas de paralelismo descritas na seção anterior, pode influenciar no desempenho das mesmas. Este modelo propõe a utilização de algumas consultas como base para verificação do desempenho dos bancos de dados. Tais consultas foram escritas com a preocupação de serem bastante semelhantes às efetuadas no dia-a-dia dos sistemas que possuem essas características.

Com o intuito de separar as consultas por características, estas foram divididas em duas categorias distintas: (i) Consultas Multidimensionais, adaptadas a partir do *Star Schema Benchmark* e detalhadamente descritas em [20], foram adaptadas para refletir as alterações efetuadas no modelo de dados utilizado no projeto CG-OLAP, e; (ii) Consultas Multidimensionais Geográficas, derivadas a partir das consultas descritas

anteriormente, porém incluindo algumas funções geográficas. Todas estas consultas estão descritas em alto nível e em linguagem SQL nos anexos I e II deste trabalho.

O banco de dados utilizado nestas consultas foi replicado nas cinco máquinas que compõem o *cluster* CG-OLAP, logo, todas as tabelas possuem a mesma quantidade de tuplas. A tabela 1 informa o total de tuplas existentes em cada uma das tabelas deste banco de dados em todas as máquinas que formam o *cluster*.

Tabela 1. Total de tuplas das tabelas do *cluster* CG-OLAP.

Tabelas	CGOLAP01	CGOLAP02	CGOLAP03	CGOLAP04	CGOLAP05
d_customer	4500000	4500000	4500000	4500000	4500000
d_geometry	5566	5566	5566	5566	5566
d_junk	12	12	12	12	12
d_part	6000000	6000000	6000000	6000000	6000000
d_supplier	300000	300000	300000	300000	300000
d_time	5000	5000	5000	5000	5000
f_lineorder	179998372	179998372	179998372	179998372	179998372

3 Ferramenta de Coleta e Análise de Dados

3.1 Introdução

O aperfeiçoamento e o desenvolvimento contínuos de componentes de redes de computadores fazem com que seja cada vez mais necessário o monitoramento e gerenciamento destas, a fim de garantir a disponibilidade dos serviços em nível de desempenho estável e aceitável aos usuários. Isto ocorre tanto em ambientes em que a rede de computadores é pequena e simples, como naqueles mais complexos ou que envolvam a adoção de uma nova tecnologia, o que pode trazer impactos não previstos e indesejáveis.

Para que seja possível a prevenção e a identificação de problemas é necessário que exista um gerenciamento adequado sobre os recursos da rede. Isto demanda investimentos em ferramentas de monitoramento do tráfego da rede que visam controlar os recursos e suas complexidades. Para se obter um gerenciamento razoável através de uma ferramenta de monitoramento é necessário que esta forneça de maneira satisfatória um bom desempenho com baixo consumo de recursos computacionais, facilidade de instalação e configuração, e interface simples e amigável. Infelizmente, a complexidade das ferramentas de monitoramento existentes está aumentando gradativamente, em razão da grande quantidade de variáveis que necessitam ser monitoradas.

Este capítulo apresenta uma ferramenta livre de monitoramento do tráfego de rede em ambientes distribuídos denominada UAS (UNIRIO Analisador de Sistemas) [9]. A UAS permite ao administrador de rede acessar informações sobre a utilização de recursos de computadores em um ambiente distribuído, através de informações consolidadas e de gráficos históricos. A UAS também atua como uma interface entre o ambiente monitorado e o NS-2.

3.2 Ferramentas Relacionadas

Algumas características são essenciais em uma ferramenta de análise de desempenho para que a mesma possa ser executada até mesmo em computadores com poucos recursos de *hardware*. A ferramenta necessita ser escrita de forma estruturada e com o mínimo de linhas de código possível, e, além de disso, é importante garantir que os processos de captura e geração de dados não causem impacto no ambiente e, desta forma não influenciem negativamente nos resultados.

A ferramenta criada nesta pesquisa, desenvolvida nas linguagens de programação PHP [21] e Bash Shell [22], tem como objetivos a monitoração remota de de computadores em uma rede local e a coleta de informações para geração de medidas de interesse e simulação de rede. Nesta seção são discutidas algumas ferramentas que possuem características semelhantes.

3.2.1 PhpSysInfo

O **PhpSysInfo** [23] é um *script open source* registrado sob licença GNU *General Public License* (GPL) [24] desenvolvido em PHP que faz um resumo do *hardware* do computador e apresenta detalhadamente no navegador informações sobre o tempo de atividade do sistema operacional, processador, memória, adaptadores de rede, utilização de disco, e dispositivos *hardware* entre outras.

Características do PhpSysInfo:

- Apresenta todas as informações coletadas de forma bastante simples e intuitiva em uma única tela divida por categorias, e estas são atualizadas automaticamente na página;
- As categorias são divididas entre “Sistema Vital”, onde são exibidas as informações básicas do sistema operacional e algumas configurações como o *hostname* e endereço IP do computador; “Informações de *Hardware*”, que exibe detalhes do processador e barramentos; “Utilização de Memória”, onde são detalhados os tipos de memórias e suas respectivas utilizações; “*Filesystems* Montados”, que exibe quais *filesystems* estão montados e seus respectivos porcentuais de utilização; e “Utilização de Rede”, que informa as interfaces existentes e a quantidade de dados trafegados por cada uma;

- Possui 7 opções de *template* e pode ser exibido em 26 línguas diferentes que podem ser alteradas em tempo real a qualquer momento;
- Um *software web server*, como Apache [25] por exemplo, e o PHP são pré-requisitos para instalação;
- Coleta informações apenas do computador onde a mesma está instalada. Não possui opção de coleta de dados de computadores remotos. Sendo assim, necessita ser instalada individualmente em todos os computadores na rede e não possibilita uma gerência consolidada dos mesmos;
- Não gera nenhum tipo de gráfico a partir dos dados coletados e não guarda evidências de informações do passado;
- Não permite a realização de testes de desempenho, apenas exibe as informações em tempo real;
- A categoria de "Utilização de Rede" é bastante sucinta e fornece apenas a quantidade de *bytes* trafegados. Não exibe nenhuma outra informação mais detalhada.

3.2.2 Cacti

O **Cacti** é uma ferramenta *open source* registrada sob licença GNU *General Public License* (GPL). Escrita quase em sua totalidade em PHP, coleta e exibe informações sobre o estado de um computador ou uma rede de computadores através de gráficos. Foi desenvolvida para oferecer flexibilidade e se adaptar a diversas necessidades com robustez. É capaz de monitorar o estado de elementos de rede e outras informações tanto do computador local quanto remoto, e cria gráficos de utilização diário, semanal, mensal e anual, onde é possível definir trechos específicos dos gráficos para que se analise um determinado período.

Características do Cacti:

- A ferramenta permite a coleta e exibição de informações do computador local e de outros computadores na mesma rede através de funcionalidade de inclusão de *hosts* remotos, e desta forma opera numa estrutura cliente-servidor.
- A coleta de informações de computadores remotos é feita através do protocolo NET-SNMP (*Simple Network Management Protocol*) [26], que deve ser previamente instalado nos mesmos.

- A ferramenta, por padrão, permite a coleta de informações e exibição em gráficos referentes ao sistema, como a carga de utilização e processos em execução; interfaces de rede; discos rígidos; processadores; partições montadas e memórias RAM e *swap*;
- Um *software web* server, o PHP, o MySQL [27] e o RRDTool (*Round Robin Database Tool*) [28] são pré-requisitos para instalação do Cacti;
- Para geração dos gráficos a aplicação utiliza a ferramenta RRDTool, que originalmente foi desenvolvida para armazenar séries de dados numéricos relativos ao estado de redes de computadores. Entretanto, também pode ser usada para o armazenamento de qualquer outra série de dados como utilização de memória, processador, etc;
- Apesar de ser bastante difundida e possuir manual de utilização e diversos tutoriais na Internet, a usabilidade da ferramenta não é simples e intuitiva, pois diversas são as configurações necessárias para que se obtenha as informações e gráficos desejados;
- Por padrão, as opções de análise de rede que a ferramenta oferece são o teste de latência através do comando *ping*, o total de dados trafegados, o total de pacotes descartados ou com erros, e o total de pacotes *unicast* ou não *unicast* trafegados;
- Apesar de ser escalável, a ferramenta não possui um agente que faça a descoberta automática ou algum utilitário para simplificar a inclusão de um *host* numa rede, ou seja, todos os dispositivos precisam ser incluídos de forma individual à partir do preenchimento manual de uma série de informações.

3.2.3 NTop

O **NTop** também é uma ferramenta *open source* registrada sob licença GNU *General Public License* (GPL), cuja função é analisar a utilização de computadores de uma rede. Ela possui uma interface amigável através de um *web browser* e oferece suporte a diversos protocolos e interfaces de rede. Os gráficos que exibem o tráfego da rede podem ser customizados pelo usuário, porém dentro de um escopo pré-definido.

Características do NTop:

- Um *software web* server, o PHP e o SQL e são pré-requisitos para instalação do NTop;
- Pode ser instalado tanto em plataformas Unix quanto *Windows*;
- Identifica e informa através de gráficos qual o computador da rede está consumindo mais a rede;
- Identifica algumas falhas de segurança mais comuns, como *portscan* e negação de serviço;
- Apesar de monitorar diversos protocolos e aspectos relacionados ao desempenho da rede, todas as parametrizações precisam ser realizadas manualmente através de linha de comando em arquivos de configuração, o que aumenta a dificuldade de utilização da ferramenta;
- É instalado como um serviço, ou seja, existe um processo que roda em *background*, o que ajuda a degradar o desempenho do mesmo, e caracteriza um ponto negativo, pois o acesso à interface é lenta mesmo quando acessada localmente.

3.3 UAS

A UAS é uma ferramenta de monitoramento que exibe informações sobre a utilização de recursos de computadores em rede e gera páginas PHP com gráficos de dados coletados a partir da utilização do protocolo NET-SNMP e da ferramenta MRTG (*Multi Router Traffic Grapher*) [29], que é utilizada como geradora dos gráficos históricos. Além disso, UAS também opera como uma interface entre um ambiente real monitorado e um ambiente simulado na ferramenta NS-2, ou seja, a ferramenta é capaz de coletar informações pertinentes à topologia da rede e seu comportamento e gerar um arquivo de saída que pode ser importado para o NS-2, e desta forma replicar automaticamente o ambiente real num ambiente simulado.

Uma vez que o ambiente real esteja replicado no ambiente simulado o usuário pode realizar testes promovendo as alterações que desejar, pois desta forma é possível estudar o comportamento deste ambiente caso o mesmo sofra alguma modificação.

É importante ressaltar que não existem relatos na literatura de ferramentas que atuem como interface entre um ambiente real e uma ferramenta de simulação de rede.

3.3.1 Características da Ferramenta UAS

UAS é uma ferramenta que utiliza a arquitetura cliente-servidor e foi desenvolvida na linguagem PHP. Portanto sua interface de interação com o usuário é o *web browser*, o que permite acesso às informações de qualquer ponto da rede.

Para a utilização da ferramenta, um dos computadores da rede deve ser eleito como servidor UAS. Ele realiza a coleta dos dados relacionados ao desempenho dos computadores remotos através do protocolo NET-SNMP, consolida estas informações e as exibe em tempo real, e gera gráficos históricos de utilização de recursos. Os demais computadores da rede são denominados clientes UAS, e apenas respondem às consultas SNMP realizadas pelo servidor UAS.

O servidor UAS necessita de um sistema operacional da plataforma Linux, um *software web server* com extensões PHP instalado, o protocolo NET-SNMP instalado e configurado com o nome de uma comunidade escolhida pelo usuário, e o MRTG. Os clientes UAS precisam apenas do protocolo NET-SNMP instalado e configurado com o mesmo nome da comunidade definida no servidor UAS.

As principais características da UAS são:

- Facilidade de gerenciamento remoto, uma vez que é possível acessar a interface de monitoramento do servidor UAS através de um *web browser* de qualquer ponto da rede;
- A baixa carga computacional exigida pela ferramenta, já que o protocolo responsável pela coleta é o NET-SNMP, um protocolo de fácil instalação amplamente difundido na Internet. Nada além deste precisa ser instalado nos clientes;
- A facilidade de inclusão de um novo cliente, que ao contrário de outras ferramentas semelhantes que exigem uma série de configurações e informações a respeito do computador remoto, no UAS é realizada através de um Bash Shell *script* que deve ser executado no servidor.

A ferramenta UAS é divida em três módulos:

- **Módulo dados -** onde são exibidas em tempo real informações gerais em modo texto sobre os clientes e a utilização de recursos como o processador, rede, memórias RAM e *swap*, disco, e dispositivos;

- **Módulo gráficos -** onde são exibidas informações em modo gráfico sobre a utilização de recursos dos clientes como o processador, rede, memórias RAM e *swap*, e disco. Estes gráficos são históricos, e armazenam informações referentes às últimas vinte e quatro horas, à última semana, ao último mês e ao último ano;
- **Módulo TCPDump *to* NS-2 -** uma interface capaz de replicar o ambiente de rede monitorado em um ambiente simulado através da conversão do *log* de saída do utilitário TCPDump em um arquivo no formato *Object Tcl* (OTcl), que é a linguagem compreendida pela ferramenta de simulação de redes NS-2. O código deste programa está descrito no anexo V deste trabalho.

A arquitetura da UAS é mostrada na figura 6. Nesta figura as setas representam o fluxo da comunicação entre os componentes da ferramenta. É importante ressaltar que *Frontend* é a parte visualizada pelo usuário, e é através deste componente que ocorre a interação da ferramenta com o mesmo.

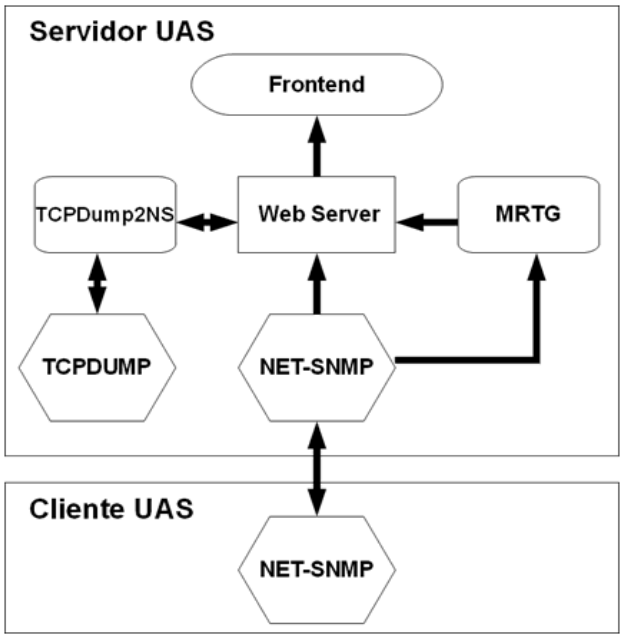

Figura 6. Arquitetura do UAS.

3.3.1.1 Pré-requisitos do UAS

O servidor UAS deve possuir sistema operacional da plataforma Linux, um *Web Server* com extensões PHP instalado e devidamente configurado, protocolo NET-SNMP habilitado e configurado com o nome da comunidade escolhida pelo usuário, e o

MRTG. Os clientes UAS necessitam apenas do protocolo NET-SNMP habilitado e configurado com o mesmo nome da comunidade definida no servidor UAS.

3.3.1.2 Instalação do UAS

A instalação do servidor UAS requer que todos os pré-requisitos sejam satisfeitos. A instalação e configuração do servidor e cliente UAS são descritas nos anexos III e IV respectivamente.

UAS é uma ferramenta livre e pode ser utilizada sem nenhum custo. É uma aplicação *Open Source* e utiliza a licença GNU *General Public License* (GPL). A figura 7 mostra a tela inicial da UAS.

Figura 7. Tela inicial do UAS.

3.3.2 Módulo Dados

O primeiro módulo do UAS é denominado **Dados**. Neste módulo a ferramenta captura informações do servidor e dos clientes UAS e exibe em modo texto informações gerais sobre os *hosts* e a utilização de seus recursos, como CPU, dispositivos, rede, memória e *filesystems*. A figura 8 exibe a seção *Sistema* da ferramenta UAS. Esta seção consolida informações básicas como *hostname*, IP,etc, de um mais computadores

clientes. A figura 9 mostra a seção *Memória*, que informa em tempo real o consumo de memórias RAM e *swap* de um ou mais computadores.

Figura 8. UAS módulo Dados – Sistema.

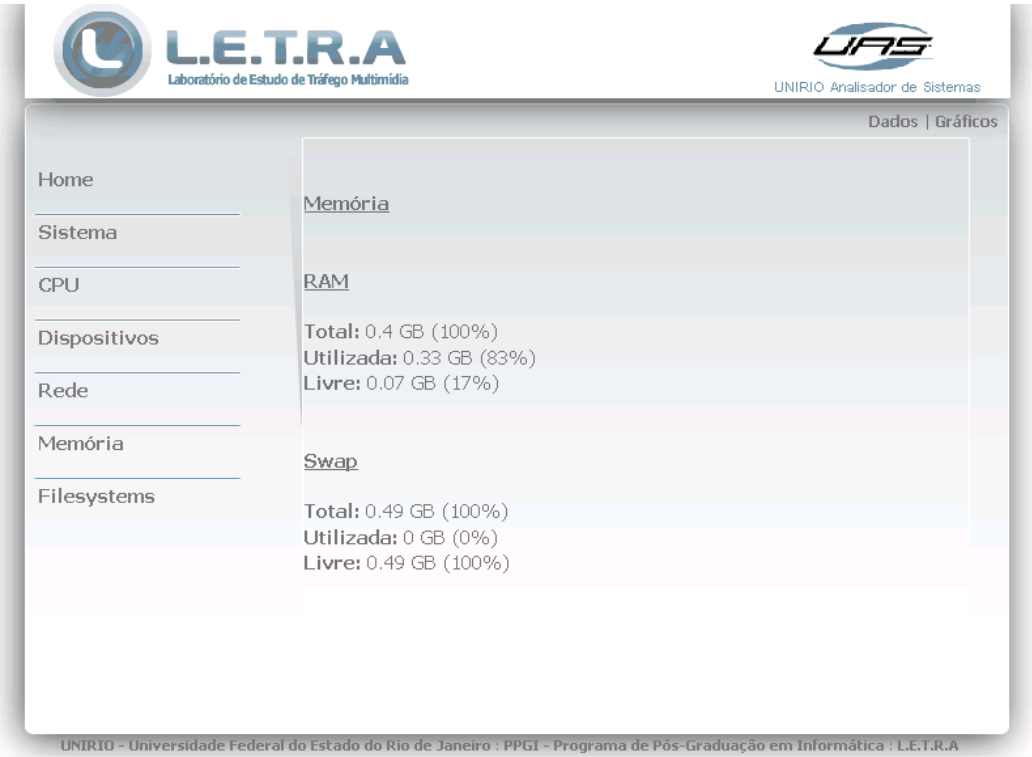

Figura 9. UAS módulo Dados – Memória.

3.3.3 Módulo Gráficos

O segundo módulo do UAS é denominado **Gráficos**. Neste módulo a ferramenta captura informações do servidor e dos clientes UAS e exibe em modo gráfico informações históricas relativas à utilização de recursos como CPU, rede, memória, *swap* e *filesystems* dos *hosts*. Os gráficos são referentes às últimas vinte e quatros horas, última semana, último mês e último ano. A figura 10 exibe a tela da ferramenta UAS que mostra os gráficos históricos de consumo de CPU das últimas vinte e quatro horas e da última semana de um computador cliente, enquanto a figura 11 exibe os gráficos históricos referentes ao consumo de memória RAM.

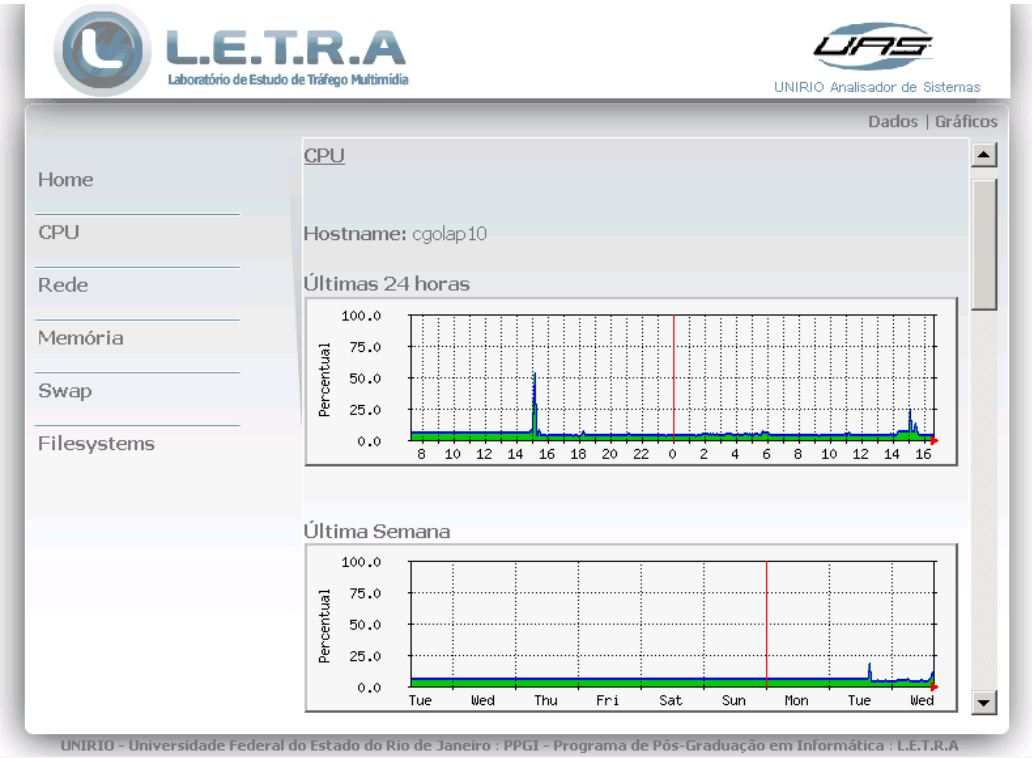

Figura 10. UAS módulo Gráficos – CPU.

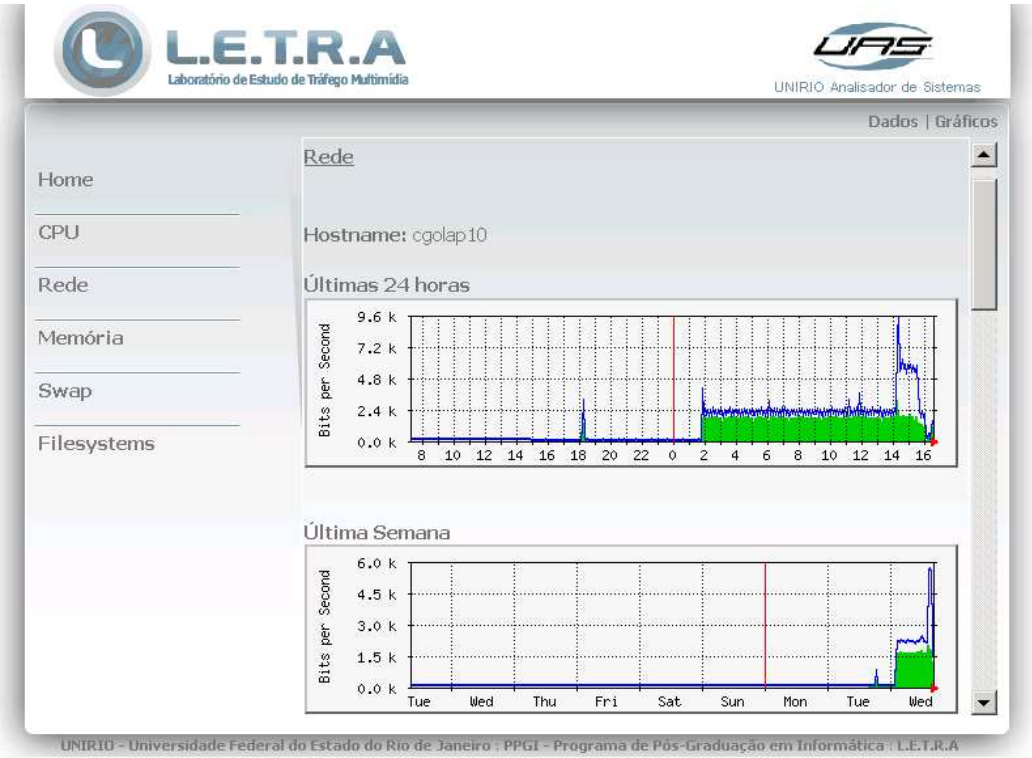

Figura 11. UAS módulo Gráficos – Rede.

3.3.4 Módulo TCPDump to NS-2

Este módulo é responsável por coletar informações de uma rede local através do utilitário TCPDump e converter o log de saída padrão em um arquivo de entrada para a ferramenta de simulação de redes NS-2. Esta ferramenta tem como finalidade o estudo do comportamento de redes e protocolos através da simulação.

A partir da importação deste arquivo no NS-2 é possível estudar o ambiente e gerar uma série de medidas de interesse relacionadas à rede. Além disso, o NS-2 também permite que sejam alteradas diversas características da rede, como por exemplo a quantidade de equipamentos, protocolos de rede e velocidade do enlace. Este recurso permite o planejamento e a análise prévia do comportamento do ambiente caso haja intenção de promover alterações no mesmo.

A figura 12 exibe a tela do módulo de conversão do arquivo de saída do TCPDump em um arquivo de entrada para o NS-2.

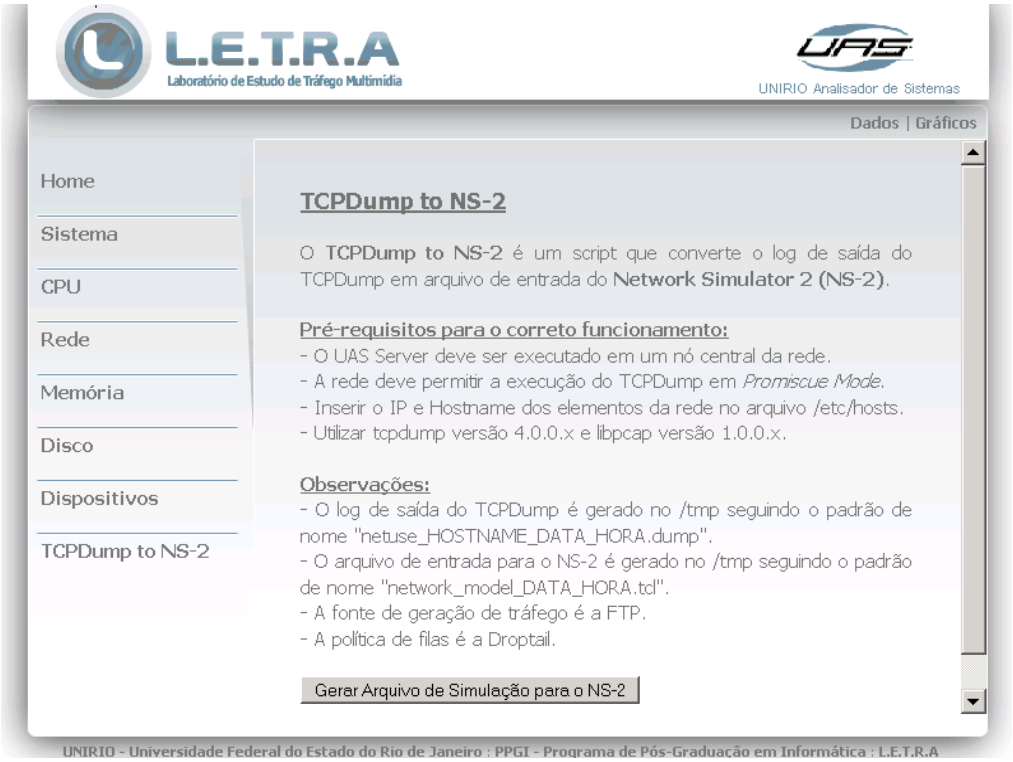

Figura 12. UAS módulo TCPDump to NS-2.

3.3.4.1 NS-2

O NS-2, ou *Network Simulator* 2, é um simulador escrito nas linguagens C++ e *Object Tcl* (Otcl) utilizado para pesquisas na área de redes. Sua concepção é baseada no conceito de simulação discreta, o que equivale a dizer que a simulação utiliza uma sequência de eventos para controlar o comportamento do modelo.

O NS-2 oferece recursos para simulações de TCP, roteamento, protocolos *multicast* sobre redes com e sem fio, dentre outros. O projeto NS começou em 1989 como uma variação do *REAL network simulator* e hoje em dia é um projeto com vida própria, mantido por desenvolvedores de diversos lugares do mundo.

O NS-2 foi a ferramenta de simulação escolhida para receber os dados coletados e tratados pelo UAS por ser capaz de representar com elevado grau de fidelidade o comportamento do tráfego de rede e ser amplamente difundido e aceito na comunidade acadêmica em todo o mundo [30].

3.3.4.2 Replicação de um Ambiente Real em um Ambiente Simulado no NS-2

A ferramenta UAS foi utilizada em um ambiente real do projeto de pesquisa CG-OLAP[3], da UNIRIO. A rede CG-OLAP contém um *cluster* de banco de dados com 4 nós. De acordo com [31], um *cluster* de banco de dados é um *cluster* que possui um SGBD (Sistema Gerenciador de Banco de Dados) em cada um dos nós, sem nenhuma modificação para adequar o comportamento dos SGBDs para o processamento em paralelo. O ambiente do projeto CG-OLAP realiza o processamento paralelo de consultas SQL sobre dados armazenados em cada nó do *cluster*. O nó central é o responsável por receber o comando SQL da consulta de uma aplicação cliente externa ao *cluster*, analisar, dividir a consulta SQL original em sub-consultas e submeter cada sub-consulta ao SGBD de cada nó do *cluster*. Os dados retornados por cada nó são enviados para o nó central, que deve compor o resultado final da consulta.

Para fins de teste, a ferramenta UAS monitorou o ambiente CG-OLAP durante 16,8 minutos, tempo de duração da execução simultânea de um conjunto de 11 consultas estipuladas no *benchmarck* TPC-H [20]. Após a coleta foi efetuada a conversão do arquivo de log do TCPDump no arquivo OTcl a ser usado no NS-2. A figura 13 mostra o ambiente de rede do *cluster* CG-OLAP replicado no ambiente simulado no NS-2.

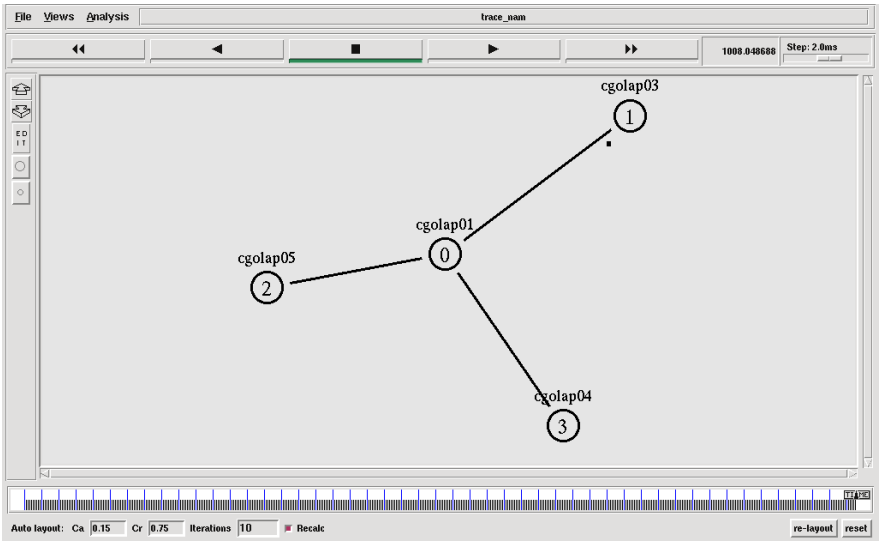

Figura 13. Ambiente de rede do *cluster* de banco de dados replicado no NS-2.

3 Financiamento FAPERJ – www.uniriotec.br/~cgolap

A retratação fiel de todos os eventos ocorridos durante um dado período de tempo permite, além da validação do ambiente, projetar e estudar o comportamento do mesmo quando submetido à alteração de configuração de algum parâmetro de rede, como a banda disponível, protocolo de congestionamento, tipo de *link*, tipo de dados trafegados, etc.

Para validação do modelo de simulação foi comparado o total de pacotes de dados e *bytes* trafegados na rede do ambiente real com o total de pacotes de dados e *bytes* trafegados no ambiente simulado durante a execução do conjunto de consultas. A figura 14 apresenta a distribuição dos pacotes trafegados em toda a rede durante o período de execução do grupo de consultas, onde fica constatado através da sobreposição dos gráficos que estes eventos, capturados pelo TCPDump, são replicados de maneira idêntica no NS-2.

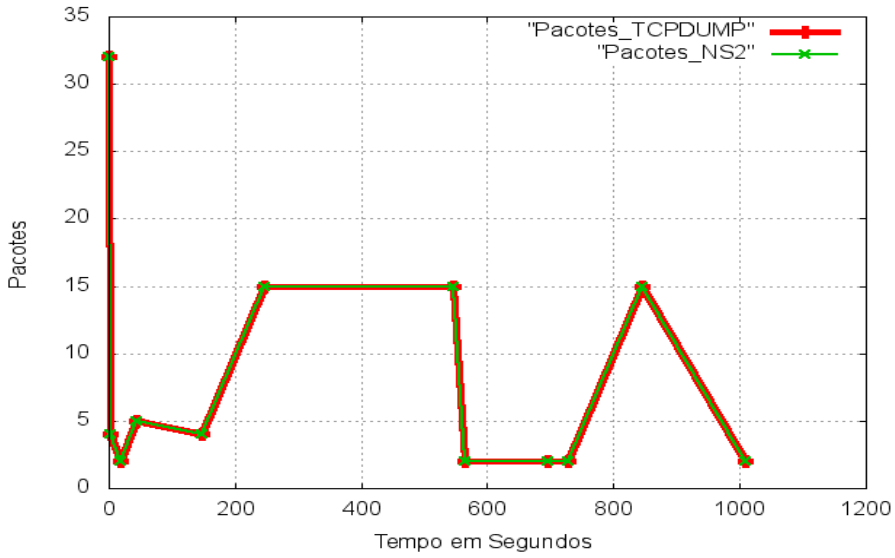

Figura 14. Total de pacotes trafegados.

A figura 15 exibe o gráfico comparativo entre o total de *bytes* contidos nos pacotes de rede durante a execução do mesmo grupo de consultas. A pequena diferença se deve ao fato dos pacotes do NS-2 serem gerados com os bytes do cabeçalho IP em relação à captura realizada pelo TCPDump, que não exibe esta informação de forma conjunta. Apesar desta diferença, é possível verificar que a distribuição dos *bytes* no ambiente simulado segue o mesmo padrão de comportamento do ambiente real, o que comprova a eficiência do **TCPDump *to* NS-2** como uma importante ferramenta nas pesquisas de análise de desempenho de redes.

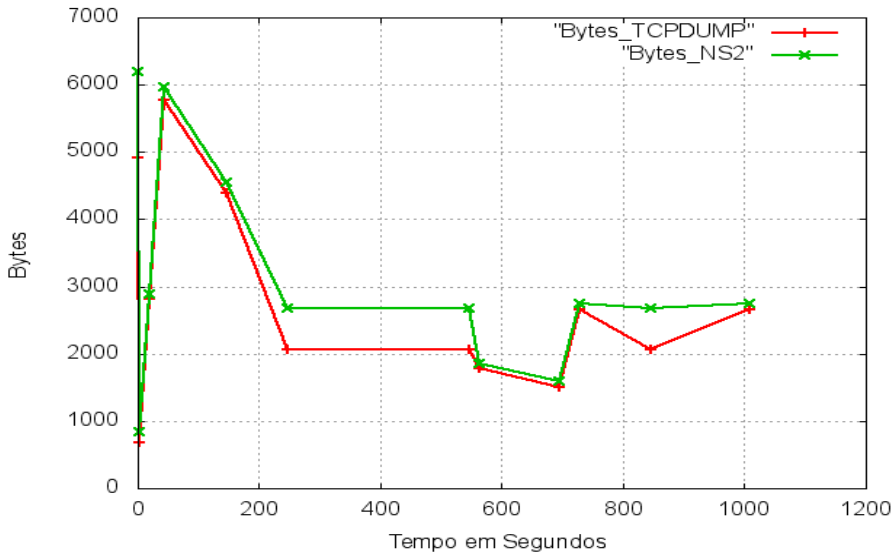

Figura 15. Total de *bytes* trafegados.

Para prever o comportamento do ambiente real através de experimentos não empíricos foram promovidas algumas alterações nos parâmetros de rede no ambiente simulado no NS-2. Estas parametrizações foram classificadas em grupos de configuração conforme exposto na tabela 2, onde foram alterados os parâmetros velocidade da rede, *Maximum Transfer Unit* (MTU), protocolo de controle de congestionamento TCP e protocolo de gerenciamento de fila de roteamento.

Tabela 2. Grupos de configuração dos parâmetros de rede alterados no NS-2.

Grupos	Configurações
1	**Velocidade da rede**: 1GB ; **MTU**: 1500 ; **Controle de congestionamento**: Tahoe ; **Gerenciamento de fila**: Droptail
2	**Velocidade da rede**: 1GB ; **MTU**: 1500 ; **Controle de congestionamento**: Tahoe ; **Gerenciamento de fila**: RED
3	**Velocidade da rede**: 1GB ; **MTU**: 1500 ; **Controle de congestionamento**: Reno ; **Gerenciamento de fila**: Droptail
4	**Velocidade da rede**: 1GB ; **MTU**: 1500 ; **Controle de congestionamento**: Reno ; **Gerenciamento de fila**: RED
5	**Velocidade da rede**: 1GB ; **MTU**: 7200 ; **Controle de congestionamento**: Tahoe ; **Gerenciamento de fila**: Droptail
6	**Velocidade da rede**: 1GB ; **MTU**: 7200 ; **Controle de congestionamento**: Tahoe ; **Gerenciamento de fila**: RED
7	**Velocidade da rede**: 1GB ; **MTU**: 7200 ; **Controle de congestionamento**: Reno ; **Gerenciamento de fila**: Droptail
8	**Velocidade da rede**: 1GB ; **MTU**: 7200 ; **Controle de congestionamento**: Reno ; **Gerenciamento de fila**: RED
9	**Velocidade da rede**: 100MB ; **MTU**: 1500 ; **Controle de congestionamento**: Tahoe ; **Gerenciamento de fila**: Droptail
10	**Velocidade da rede**: 100MB ; **MTU**: 1500 ; **Controle de congestionamento**: Tahoe ; **Gerenciamento de fila**: RED
11	**Velocidade da rede**: 100MB ; **MTU**: 1500 ; **Controle de congestionamento**: Reno ; **Gerenciamento de fila**: Droptail
12	**Velocidade da rede**: 100MB ; **MTU**: 1500 ; **Controle de congestionamento**: Reno ; **Gerenciamento de fila**: RED

O ambiente de rede real do *cluster* CG-OLAP está configurado com os parâmetros descritos no grupo 3 da tabela 2.

A tabela 3 exibe, para cada um dos grupos definidos na tabela 2, as medidas de interesse extraídas a partir da simulação.

Tabela 3. Medidas de interesse obtidas através da simulação no NS-2.

	Grupo 1	Grupo 2	Grupo 3	Grupo 4	Grupo 5	Grupo 6	Grupo 7	Grupo 8	Grupo 9	Grupo 10	Grupo 11	Grupo 12
Total de Pacotes Trafegados	1200	1200	1200	1200	1200	1200	1200	1200	1200	1200	1200	1200
Média de Pacotes por Segundo	1,19	1,19	1,19	1,19	1,19	1,19	1,19	1,19	1,19	1,19	1,19	1,19
Total de Bytes Trafegados	148224	148224	148224	148224	148224	148224	148224	148224	148224	148224	148224	148224
Média de Bytes por Segundo	147,04	147,04	147,04	147,04	147,04	147,04	147,04	147,04	147,04	147,04	147,04	147,04
Tamanho Médio dos Pacotes (em Bytes)	123,52	123,52	123,52	123,52	123,52	123,52	123,52	123,52	123,52	123,52	123,52	123,52
Total de Pacotes de Dados Trafegados	100	100	100	100	100	100	100	100	100	100	100	100
Média de Pacotes de Dados por Segundo	0,09	0,09	0,09	0,09	0,09	0,09	0,09	0,09	0,09	0,09	0,09	0,09
Total de Bytes Trafegados em Pacotes de Dados	37408	37408	37408	37408	37408	37408	37408	37408	37408	37408	37408	37408
Média de Bytes por Segundo em Pacotes de Dados	37,11	37,11	37,11	37,11	37,11	37,11	37,11	37,11	37,11	37,11	37,11	37,11
Tamanho Médio dos Pacotes de Dados (em Bytes)	374,08	374,08	374,08	374,08	374,08	374,08	374,08	374,08	374,08	374,08	374,08	374,08
Pacotes Descartados	0	0	0	0	0	0	0	0	0	0	0	0
*** Throughput Médio em Bps**	124	124	124	124	124	124	124	124	124	124	124	124
*** Latência Média**	0.000019	0.000019	0.000019	0.000019	0.000019	0.000019	0.000019	0.000019	0.000134	0.000134	0.000134	0.000134

Baseado nos resultados obtidos através da simulação foi constatado que o ambiente real manterá o mesmo comportamento sem ganho ou perda significativa de desempenho ainda que a configuração destes parâmetros sejam alteradas. Isso ocorre devido às características inerentes ao *cluster* de banco de dados instalado neste ambiente, que conforme observado não gera volume de tráfego intenso quando envia consultas e recebe as respostas dos nós participantes. Desta forma, é possível afirmar que até mesmo uma rede que possua configurações inferiores àquelas utilizadas pelo ambiente analisado suportará o mesmo de maneira satisfatória.

3.3.4.3 Simulação de Inclusão e Remoção de Nós do *Cluster*

A inclusão e remoção de nós de uma rede é uma tarefa corriqueira no âmbito da administração de redes, portanto, é de extrema importância a realização de testes sobre o comportamento de um ambiente quando este sofre alteração na quantidade de nós existentes.

Uma vez que a rede do *cluster* CG-OLAP possui apenas cinco máquinas, sendo um nó mestre e quatro escravos, a validação do tempo total de execução de um conjunto de consultas SQL baseadas no ambiente simulado pode ser realizada apenas para configurações de 2, 3 e 4 nós escravos. Portanto, para validar o ambiente de simulação no NS-2 foi executado no ambiente real o mesmo conjunto de consultas descritos na seção 3.3.4.2 para configurações de *cluster* com 2, 3 e 4 nós, porém, para a validação do tempo total de execução nos eventos de inclusão e remoção de nó do *cluster* foram considerados os tempos médios de 90 execuções consecutivas deste conjunto de consultas. A repetição das consultas foi realizada a fim de reduzir os efeitos de

flutuações momentâneas de carga nas máquinas em razão da execução simultânea inevitável de outros processos, como os do próprio sistema operacional por exemplo.

Para testar o comportamento do ambiente simulado quando há eventos de inclusão e remoção de nós de uma rede foi utilizado o **TCPDump2NS** para replicar no NS-2 o comportamento do ambiente CG-OLAP quando executado o conjunto de consultas com 3 nós escravos ativos. A partir desta configuração foi simulado o comportamento do ambiente com 2 e 4 nós escravos. Em seguida foram comparados os tempos totais obtidos no ambiente simulado com os observados no ambiente real. A figura 16 exibe a comparação dos tempos totais de execução do conjunto de consultas SQL nos ambientes simulado e real.

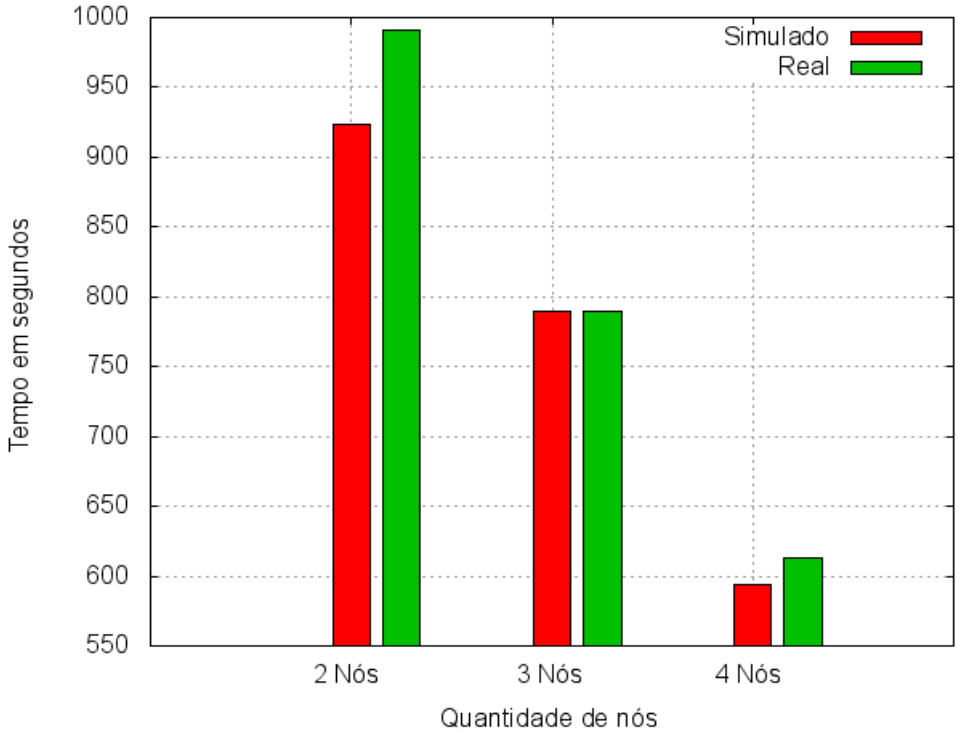

Figura 16. Tempo total de execução de consultas no *cluster* CG-OLAP.

A partir desta validação foi simulada a remoção e a inclusão de um nó no *cluster*. A configuração com 2 nós apresentou uma redução de 6,79% (67,3 segundos) no tempo total de execução, enquanto com 4 nós a diferença foi de 3,24% (19,9 segundos). Este resultado é coerente com o ambiente CG-OLAP, já que nos testes realizados foi verificado que a variância no tempo de resposta é em torno de 10%.

É possível, portanto, concluir, que a ferramenta fornece uma boa aproximação para testar ambientes que não existem a partir do ambiente monitorado.

4 Modelos de Fragmentação de Dados

4.1 Introdução

A avaliação de desempenho é uma das tarefas mais importantes no processo de desenvolvimento de novos sistemas de comunicação e computação, pois novas tecnologias e algoritmos podem ser propostos e avaliados antes de serem implementados. A modelagem e a simulação são exemplos de técnicas que compõem a avaliação de desempenho.

Um modelo é uma réplica ou uma abstração da característica essencial de um processo, ou seja, é uma visão simplificada de um sistema, onde se busca a realidade do mesmo de maneira a fornecer diagnósticos precisos dentro de um nível aceitável de tolerância. Um modelo é frequentemente a única alternativa prática se o sistema ainda não existe ou se é necessário analisar o sistema submetido a uma carga ainda inexistente [32].

O desenvolvimento de um modelo requer: a escolha de um nível de abstração que deverá ser utilizado para descrever o sistema a ser modelado; a definição das características do sistema que serão incluídas no modelo; a decisão dos índices de desempenho apropriados que serão observados; e o ajuste dos valores numéricos como parâmetros do sistema.

A simulação é uma técnica que tem como objetivo imitar o comportamento de um sistema real. Pode ser considerada como a reprodução funcional de uma realidade através de um modelo a ser analisado e avaliado. Ela deve descrever as características funcionais do modelo e conter todos os detalhes relevantes. Simulação é a forma mais popular de avaliar sistemas reais, pois oferecem grande facilidade para ser empregada, uma vez que as operações e carga são descritas através de algoritmos apropriados em

programas de computador. São implementados em linguagens próprias para as construções destes modelos e os índices de desempenho são obtidos através da monitoração do programa em execução [33]. Um programa de simulação, por exemplo, deve gerar eventos de chegadas e serviço, manter o estado da fila, calcular e gerar medidas de interesse.

Um modelo matemático é um conjunto de equações que se apoiam sobre o sistema real e representam as hipóteses utilizadas na construção do modelo de maneira quantitativa. Estas equações são resolvidas tendo como base valores conhecidos ou previstos pelo sistema real, e são avaliadas por meio da comparação com as medidas obtidas no ambiente real. A modelagem matemática pode reduzir a necessidade de realização de experimentos no ambiente real, pois o modelo experimental pode fornecer respostas a questões particulares baseadas em hipóteses alternativas.

Em muitas situações são usados modelos estocásticos[4] para analisar o desempenho de um sistema real. Modelos estocásticos possuem comportamento probabilístico, ou seja, seu comportamento é não determinístico. Cada estado desse processo não determina completamente qual será o seu próximo estado. Um processo cuja probabilidade de transição do fenômeno depende apenas do estado atual e do próximo é denominado processo de Markov. Uma sequência de estados que segue este processo é denominada cadeia de Markov.

Cadeias de Markov podem ser empregadas para determinar as probabilidades de transição entre estados que evoluem de maneira estocástica, pois conhecendo apenas o estado atual, e não sua trajetória, é possível prever os estados de um processo em andamento.

Existem dois grupos de modelos markovianos com espaço de estados discretos: o de tempo discreto, aquele onde as mudanças são realizadas apenas em instantes discretos de tempo, e o de tempo contínuo,aquele em que o estado do sistema pode mudar em qualquer instante de tempo.

No capítulo 3, o simulador NS-2 foi utilizado para estudar o comportamento do ambiente CG-OLAP quando mudanças são introduzidas na infraestrutura. Neste capítulo, modelos markovianos são usados para caracterizar consultas SQL com as técnicas de fragmentação virtual de dados **SVP** (*Simple Virtual Partitioning*) [1], **AVP**

4 Família de variáveis aleatórias indexadas por elementos *t* pertencentes a determinado intervalo temporal.

(*Adaptive Virtual Partitioning*) [3] e **AVP_WR** (*Adaptive Virtual Partitioning – Workload Redistribution*) [3] discutidas no capítulo 3.

4.2 Processos Markovianos

Processo markoviano é um tipo de processo estocástico que possui limitada dependência em relação ao seu passado. A propriedade, ou memória, markoviana, significa que os estados anteriores não influenciam na predição dos estados posteriores desde que o estado atual seja conhecido. Uma cadeia de Markov é uma sequência X_1, X_2, X_3, ... de variáveis aleatórias, onde o conjunto de valores que essas variáveis podem assumir é chamada de espaço de estados ***S***.

Seja X_n o estado do processo no tempo *n*. Se a distribuição de probabilidade condicional de X_{n+1} nos estados passados é uma função apenas de X_n, então,

$$Pr(X_{n+1}=x|X_0, X_1, X_2, \ldots, X_n)=Pr(X_{n+1}=x|X_n). \tag{4.1}$$

Este processo é denominado *memoryless* (sem memória) porque a probabilidade do próximo estado depende apenas do estado atual. Logo, a evolução do processo Markoviano em um tempo futuro, condicionado no presente e no passado, somente depende do seu valor no presente [34].

4.2.1 Cadeias de Markov de Tempo Discreto

Numa cadeia de Markov de tempo discreto as transições ocorrem em instantes definidos onde o tempo de permanência em um estado é geometricamente distribuído.

Por exemplo, considere uma cadeia de Markov com três estados, cuja probabilidade de transição do estado $i \in \boldsymbol{S}$ no instante $t = n$ definida como $p_{ij} = \boldsymbol{P}[X_n = j \,|X_{n-1} = i]$. A matriz de probabilidades de transição ***P*** desta cadeia é representada por:

$$P = \begin{bmatrix} p_{11} & p_{12} & p_{13} \\ p_{21} & p_{22} & p_{23} \\ p_{31} & p_{32} & p_{33} \end{bmatrix} \tag{4.2}$$

A soma de cada linha da matriz é igual a 1. Além disso, cada linha também representa a probabilidade de transição de um estado para outro em um único passo.

Iniciando em qualquer dos estados, em um passo, a partir da matriz de probabilidades de transição, é possível definir a probabilidade de uma cadeia de Markov de tempo discreto em alcançar o estado $j \in \boldsymbol{S}$ no *n*-ésimo passo usando a equação de *Chapman-Kolmogorov* [35]:

$$p_{i,j}(n) = \sum_{k \in S} p_{i,k}(m)\, p_{k,j}(n-m), 0 \leqslant m \leqslant n,$$

(4.3)

onde a transição do estado *i* para o *k* ocorre em *m* passos e a transição do estado *k* para o *j* ocorre em *n* - *m* passos.

Seja $\pi^{(n)} = (\pi_0^{(n)}, \pi_1^{(n)}, ...)$ o vetor de probabilidades cujo elemento $\pi_i^{(n)}$ representa a probabilidade da cadeia de Markov de tempo discreto estar no estado *i* no passo *n* [36]. Então, é possível definir:

$$\begin{gathered}
\pi^{(1)} = \pi^{(0)} P \\
\pi^{(2)} = \pi^{(1)} P = \pi^{(0)} P^2 \\
\vdots \\
\pi^{(n)} = \pi^{(n-1)} P = \pi^{(0)} P^n \\
\vdots \\
\pi = \pi P
\end{gathered}$$

(4.4)

A equação 4.4 mostra que as características dos estados de uma cadeia de Markov irredutível[5], aperiódica e não nula se mantém estáveis quando $n \rightarrow \infty$. Os valores obtidos para π são então conhecidos como probabilidade estacionária[6].

4.2.2 Cadeias de Markov de Tempo Contínuo

Em uma cadeia de Markov de tempo contínuo o processo é transicionado de um estado para outro após um período de tempo exponencialmente distribuído, e a transição de um estado para outro depende apenas do estado atual.

5 Cadeia onde um estado pode ser alcançado a partir de qualquer outro estado.
6 Processo aleatório onde as suas propriedades estatísticas não variam com o tempo.

Para um modelo com três estados, a matriz de taxa de transição entre os estados é representada por:

$$Q = \begin{bmatrix} q_{11} & q_{12} & q_{13} \\ q_{21} & q_{22} & q_{23} \\ q_{31} & q_{32} & q_{33} \end{bmatrix},$$

(4.5)

onde q_{ij} corresponde à taxa de transição do estado i para o estado j , $1 \leq i$, $j \leq 3$, e

$$q_{ij} = \sum_{j=1,\, i \neq j}^{3} q_{ij}.$$

O vetor de probabilidades do estado estacionário de uma cadeia de Markov de tempo contínuo é obtido por:

$$\pi Q = 0.$$

(4.6)

De acordo com [37], é possível obter um processo discreto a partir de um processo markoviano de tempo contínuo usando o método de uniformização.

4.2.3 Método de Uniformização

Considere uma cadeia de Markov X de tempo contínuo e espaço de estados finito com **M** estados. Seja ***Q*** a matriz de taxas de transição de X, é possível transformar estas taxas em uma cadeia Z de tempo discreto e com matriz ***P*** de probabilidades de transição da seguinte forma:

$$P = I + \frac{Q}{\Lambda}$$

(4.7)

onde Λ é a taxa de uniformização e é escolhida tal que seja maior ou igual à maior taxa de saída dentre todos os estados de X.

4.3 Ferramentas de Modelagem

Existem diversas ferramentas que auxiliam na tarefa de definição e análise de modelos. Algumas são aplicáveis apenas a tipos específicos de modelos como a SAVE (*System Availability Estimator*) [38], enquanto outras são mais genéricas, como o TANGRAM-II [11].

Os modelos propostos neste trabalho foram desenvolvidos no TANGRAM-II em razão dos mesmos poderem ser representados através de modelos markovianos, e também porque esta ferramenta oferece recursos satisfatórios para extração de medidas de interesse.

4.3.1 TANGRAM-II

A ferramenta TANGRAM-II é um ambiente de modelagem desenvolvido para fins de pesquisa e ensino que provê uma interface flexível para descrição de sistemas de computação e comunicação [39]. O paradigma de modelagem utilizado pelo TANGRAM-II foi proposto em [11]. Neste paradigma, o sistema a ser modelado é representado por uma coleção de objetos que interagem entre si através do envio de mensagens. O estado interno de um objeto é representado por um conjunto de variáveis. O comportamento de um objeto é definido por eventos que possuem uma condição e um conjunto de ações, e também por ações que estão associadas ao recebimento de mensagens [39]. Para que um novo evento seja gerado por um objeto é necessário que alguma condição associada ao mesmo seja satisfeita. Estes eventos são disparados obedecendo alguma distribuição temporal, como exponencial ou determinística por exemplo.

A comunicação entre os objetos é realizada por meio de troca de mensagens, que quando entregues ao destinatário executam alguma ação associada, que por sua vez, pode possuir uma determinada probabilidade.

4.4 Modelos Markovianos

Um dos objetivos desta pesquisa é a definição de modelos de três técnicas de fragmentação virtual de dados para consultas SQL em *clusters* de bancos de dados

distribuídos. A primeira técnica discutida é a SVP [1], que originou as outras duas, AVP [3] e AVP_WR [3].

O modelo SVP apresentado neste capítulo foi feito a partir de trabalhos publicados na literatura, enquanto os modelos AVP e AVP_WR foram definidos a partir dos resultados obtidos por [3] e de experimentos empíricos realizados no ambiente de pesquisa do projeto CG-OLAP.

Os modelos fazem a suposição que uma consulta SQL pode ser representada por uma carga de trabalho **W**. A quantidade de trabalho destinada a um determinado nó por consulta corresponde à carga de trabalho individual do mesmo, enquanto o somatório do trabalho realizado por todos os nós corresponde à carga total de trabalho do sistema.

Suponha, por exemplo, que uma consulta SQL com carga de trabalho total igual a 64 é submetida a um banco de dados replicado em quatro nós. Se a fragmentação dos dados estiver uniformemente distribuída então cada nó receberá uma carga de trabalho igual à 16.

4.4.1 Modelo SVP

No modelo SVP, um dos nós do *cluster* é responsável por dividir e enviar a consulta SQL em tantas sub-consultas, fragmentos virtuais, quantos forem o número de nós deste *cluster*. O objetivo da divisão é distribuir de forma equivalente entre os nós do *cluster* a carga de trabalho.

O modelo é composto por *k* objetos: um objeto é denominado **nó mestre** e é responsável pela distribuição da carga de trabalho para os outros *k* - 1 objetos, chamados **nós escravos**. A carga de trabalho é representada por um vetor com *k* posições, onde uma posição define a carga total de trabalho ainda não processada no sistema, e as outras posições correspondem à carga de trabalho ainda a ser processada por cada nó escravo. Note que a soma das cargas individuais dos nós escravos corresponde a carga total de trabalho solicitada pelo nó mestre. O algoritmo do modelo SVP está descrito no anexo VI deste trabalho.

Portanto, o modelo é caracterizado por *k* variáveis de estado, uma para cada objeto do sistema. A figura 17 apresenta o diagrama de transição de estados do modelo SVP com apenas dois nós escravos e carga de trabalho igual a seis. Neste exemplo o

estado 1 caracteriza o período de inatividade, enquanto os estados de 2 a 15 indicam que os nós escravos estão processando carga de trabalho demandada pelo nó mestre.

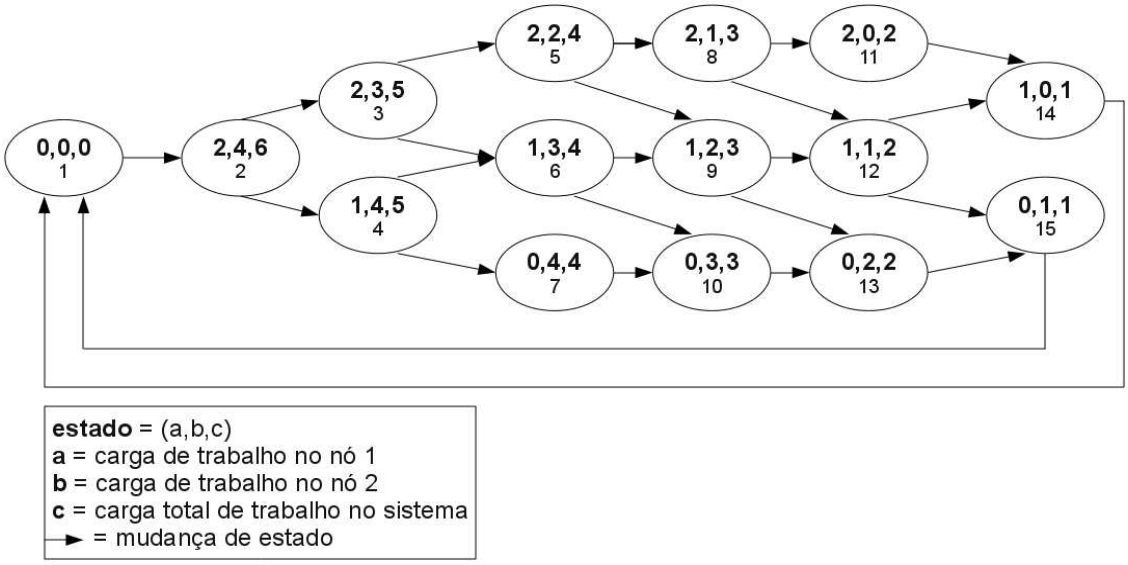

Figura 17. Modelo SVP com 2 nós escravos e carga de trabalho igual a 6.

O principal problema desse modelo é a possibilidade de um grande número de estados quando o número de nós do *cluster* é grande. Por exemplo, para um *cluster* com 4 nós e **W**=512 uniformemente distribuído, a cadeia de Markov é composta por mais de 100.000 estados.

Se considerarmos que os nós possuem a mesma capacidade de processamento, ou seja, usam a mesma taxa de serviço, é possível representar a figura 17 por um servidor Erlang [36] de seis estágios, como mostra a figura 18.

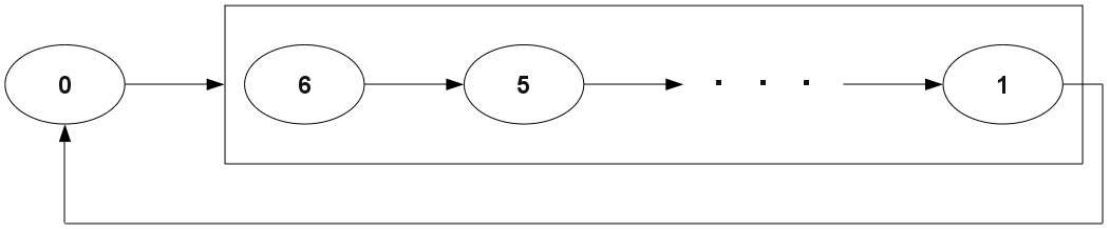

Figura 18. Servidor Erlang com 6 estágios.

O tempo médio de serviço de uma distribuição Erlang de seis estágios é dada por:

$$\bar{X} = 6.\frac{1}{\mu} \tag{4.8}$$

onde μ é a taxa de serviço de cada nó.

Além disso, a matriz de taxas de transição do modelo possui uma estrutura quase diagonal. A figura 19, gerada pelo TANGRAM-II, exibe a matriz de transição da cadeia de Markov mostrada na figura 17. É possível resolver este tipo de modelo usando a modificação da eliminação Gaussiana clássica proposta em [40].

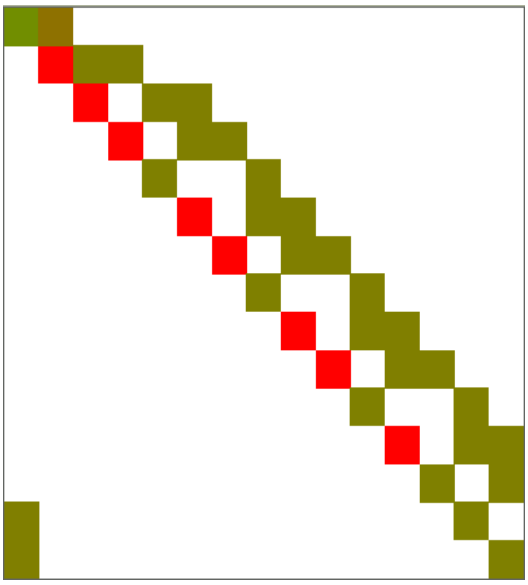

Figura 19. Matriz de taxas de transição com estrutura quase diagonal.

É possível resolver os modelos desenvolvidos utilizando Erlang em razão das máquinas serem homogêneas. Se as máquinas possuíssem arquitetura e, consequentemente taxas de serviço heterogêneas, seria necessário encontrar o estado estacionário da cadeia de Markov para calcular o tempo médio de retorno ao estado zero.

4.4.1.1 Comparação de Resultados entre SVP Real e SVP Modelo

A comparação dos resultados obtidos por [3] com os resultados da modelagem analítica do modelo SVP para *clusters* com 2, 4, 8, 16, 32 e 64 nós foi realizada usando as consultas SQL Q21 e Q1 descritas em [3], sendo que a primeira tem carga de trabalho uniformemente balanceada, e a segunda possui distorção de dados, ou seja, carga de trabalho desbalanceada. A figura 20 mostra a distorção de dados utilizada para a consulta Q1 no modelo. Nesta figura são exibidas as diferentes cargas de trabalho definidas para cada nó escravo em um *cluster* de 64 nós. Esses valores serviram como base para a geração das cargas de trabalho das outras configurações de *cluster*. Por

exemplo, para produzirmos tamanhos a serem utilizados em experimentos com 32 nós, foram somados 2 tamanhos consecutivos; para 16 nós, 4 consecutivos; e assim sucessivamente. Este desbalanceamento foi utilizado em todos os experimentos com distorção de dados.

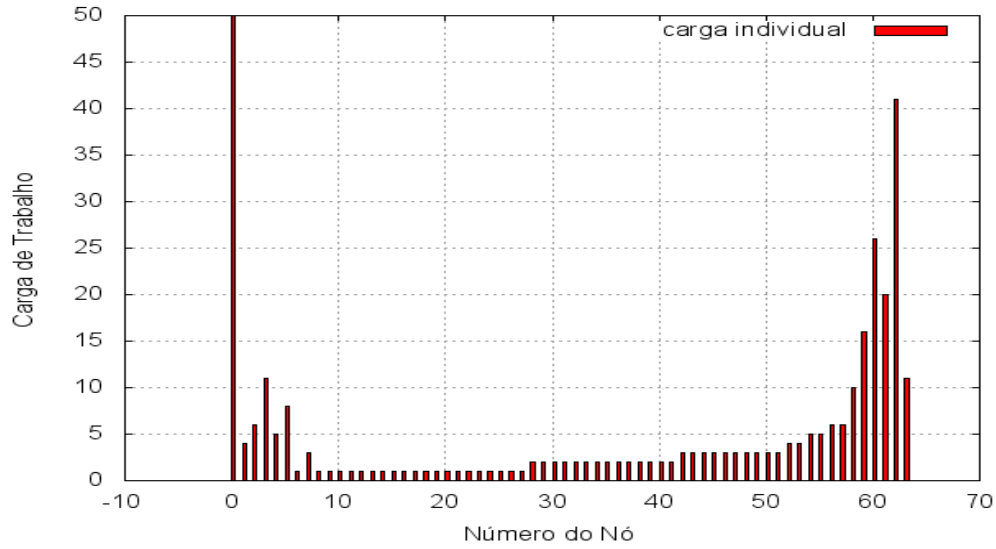

Figura 20. Carga de trabalho utilizada nos experimentos com distorção de dados.

As figuras 21 e 22 exibem os gráficos com os tempos de execução normalizados em 100 obtidos por [3] e da modelagem analítica sobre o modelo SVP realizadas nesta pesquisa.

A figura 21 mostra que os resultados obtidos a partir do modelo SVP para uma consulta SQL executada em um banco de dados uniformemente distribuído são bastante semelhantes aos obtidos nos testes empíricos realizados em [3]. A configuração de *cluster* com 2 nós no ambiente real apresentou um acréscimo de 0,46% de tempo em relação à simulação realizada com o modelo SVP, enquanto os testes com 4, 8 e 64 nós apresentaram, respectivamente, acréscimo de 1,05%, 0,92% e 0,11%. Para as consultas realizadas com 16 e 32 nós foram observados, respectivamente, um acréscimo de 0,41% e 0,03% de tempo em favor dos testes realizados com o modelo SVP.

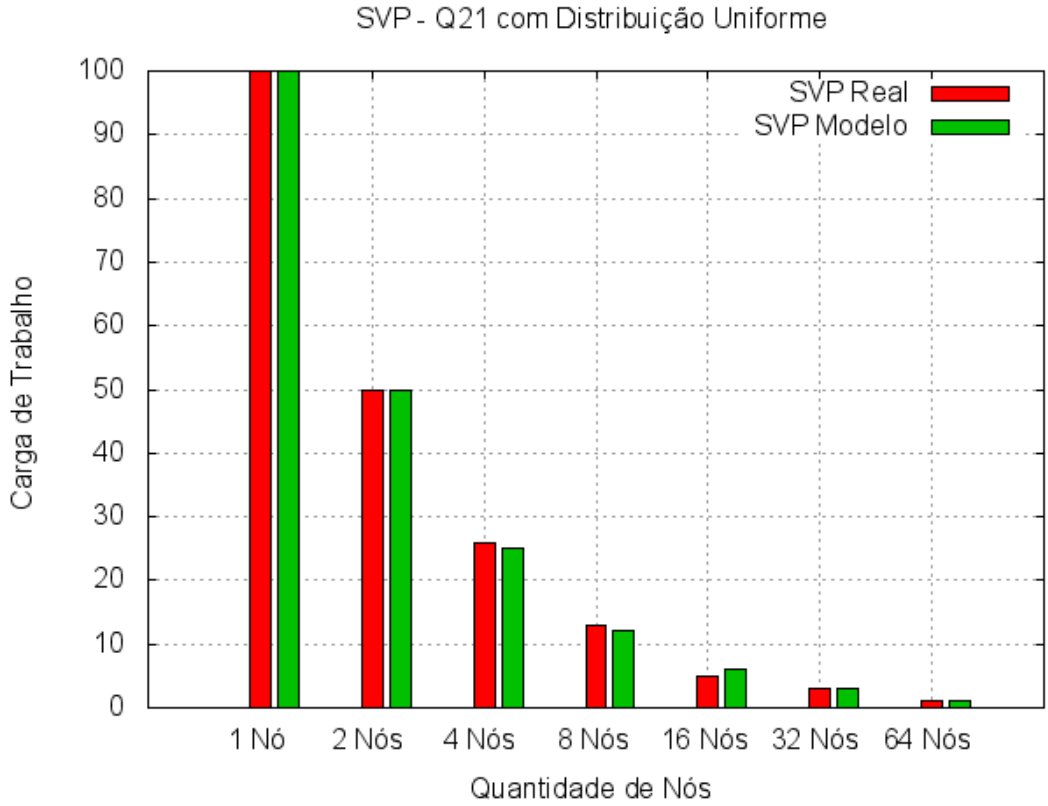

Figura 21. Consulta SQL com distribuição uniforme de dados.

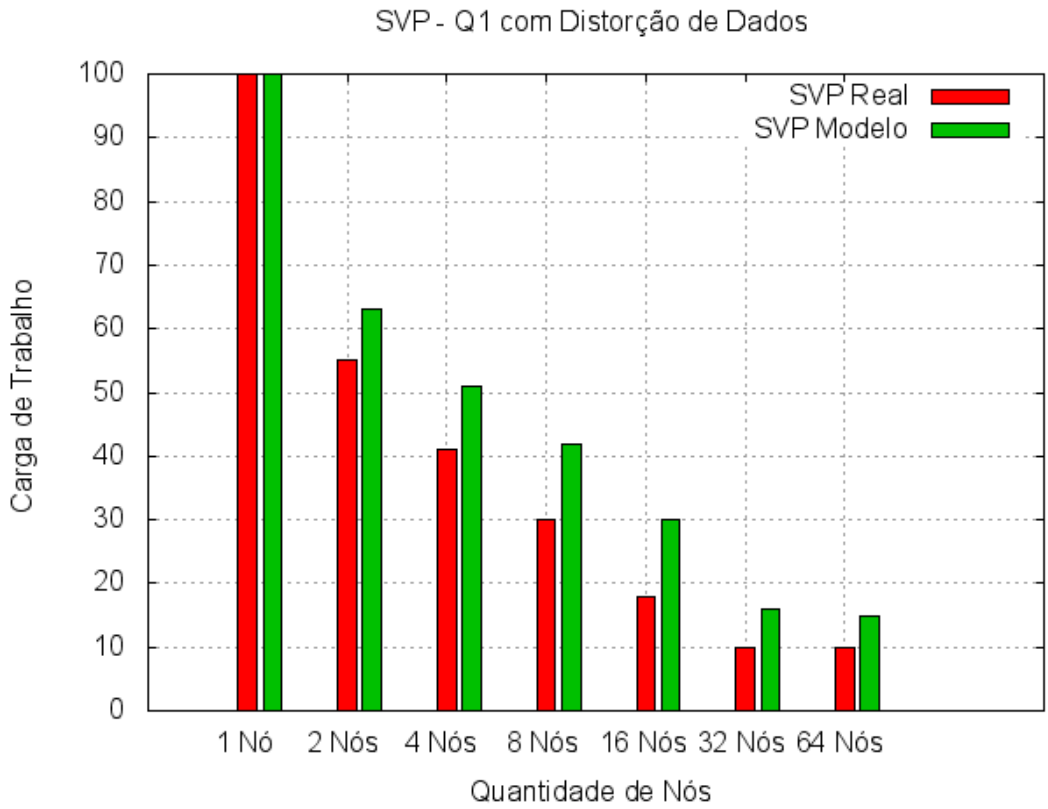

Figura 22. Consulta SQL com distorção de dados.

A figura 22 mostra os resultados obtidos a partir do modelo e do ambiente real de uma consulta SQL com fragmentação SVP executada em um banco de dados com carga de trabalho desbalanceada em *clusters* com 2, 4, 8, 16, 32 e 64 nós. Foi observado que o tempo de execução do modelo é superior ao do ambiente real com variação entre 5,61% (64 nós) e 12,34% (16 nós) para todas as configurações de *cluster* utilizadas. A diferença de tempo verificada entre os dois ambientes pode ser justificada pelo fato da definição da carga de trabalho usada no modelo ter sido obtida a partir do gráfico de distorção de dados apresentado em [3]. É importante ressaltar que o trabalho de [3] não especifica a variância do tempo de resposta nos testes feitos, portanto, é possível assumir o valor de 10% como observado no ambiente CG-OLAP. Além disso, esta diferença também se deve ao fato dos modelos não contemplarem o processo de concatenamento das respostas das sub-consultas e envio do resultado final à uma aplicação cliente.

4.4.2 Modelo AVP

O modelo AVP opera da mesma maneira que o SVP, entretanto, este algoritmo trabalha com maior velocidade que o SVP em situações onde há distorção de dados. Isso ocorre pelo fato desta técnica dividir a consulta em fragmentos menores, a granularidade fina, conforme descrito no capítulo 3. Esta característica pode ser representada no modelo pelo aumento da taxa de serviço. Por exemplo, a taxa de serviço da consulta Q21 é 20,5% mais rápida no AVP quando comparada ao SVP. O algoritmo do modelo AVP está descrito no anexo VII deste trabalho.

A definição dos objetos e o diagrama de transição de estados também possuem o mesmo comportamento do modelo SVP. Por esta razão não são repetidos nesta seção.

4.4.2.1 Comparação de Resultados entre AVP Real e AVP Modelo

Assim como no modelo SVP, foi realizada a comparação dos resultados obtidos por [3] com os resultados da simulação do modelo AVP para *clusters* com 2, 4, 8, 16, 32 e 64 nós. As consultas utilizadas também foram as mesmas, SQL Q21, com carga de trabalho balanceada, e Q1, com carga desbalanceada, cuja distribuição de carga é a mesma apresentada na figura 20.

As figuras 23 e 24 mostram os gráficos com os tempos de execução normalizados em 100 obtidos por [3] e das modelagens analíticas sobre o modelo AVP realizadas nesta pesquisa.

A figura 23 mostra que os resultados obtidos a partir do modelo AVP para uma consulta SQL executada em um banco de dados uniformemente distribuído são bastante semelhantes aos obtidos nos testes empíricos realizados em [3]. A configuração de *cluster* com 2 e 4 nós no ambiente modelado apresentaram respectivamente um acréscimo de 1,99% e 0,12% de tempo em relação aos testes realizados no ambiente real AVP. Nas configurações de *cluster* com 8, 16, 32 e 64 nós o ambiente real AVP apresentou ligeiro acréscimo de tempo em relação ao modelo. As diferenças observadas foram 0,75%, 1,03%, 0,51% e 0,34% respectivamente.

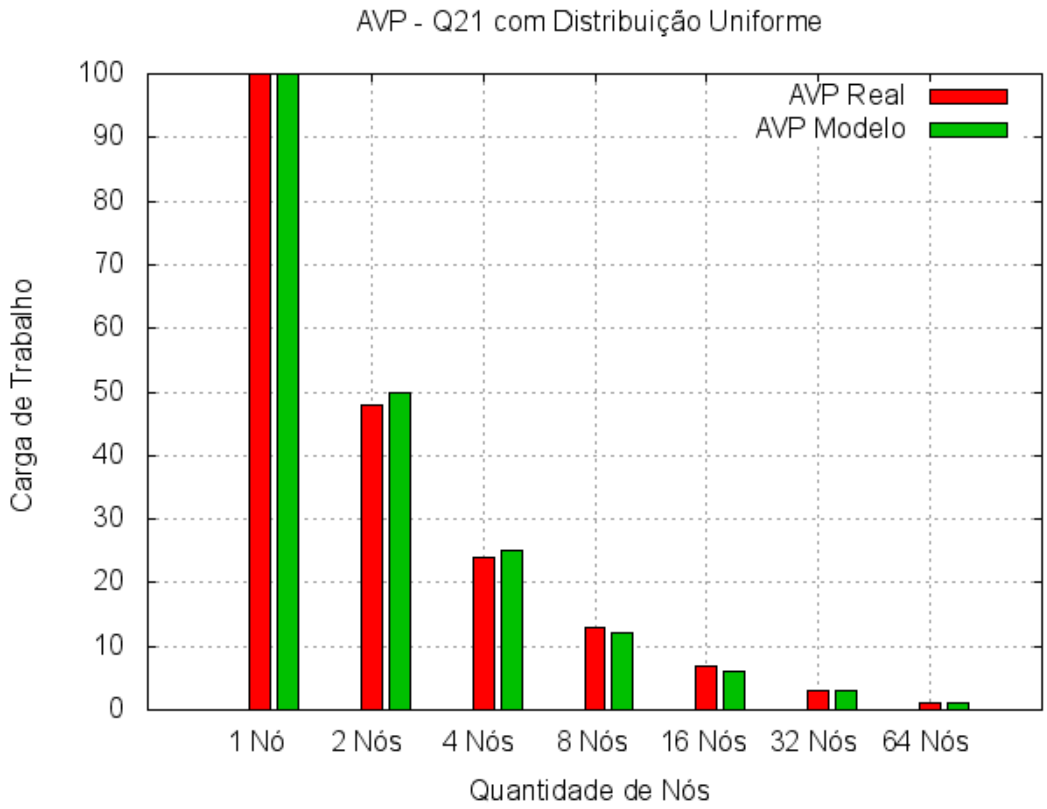

Figura 23. Consulta SQL com distribuição uniforme de dados.

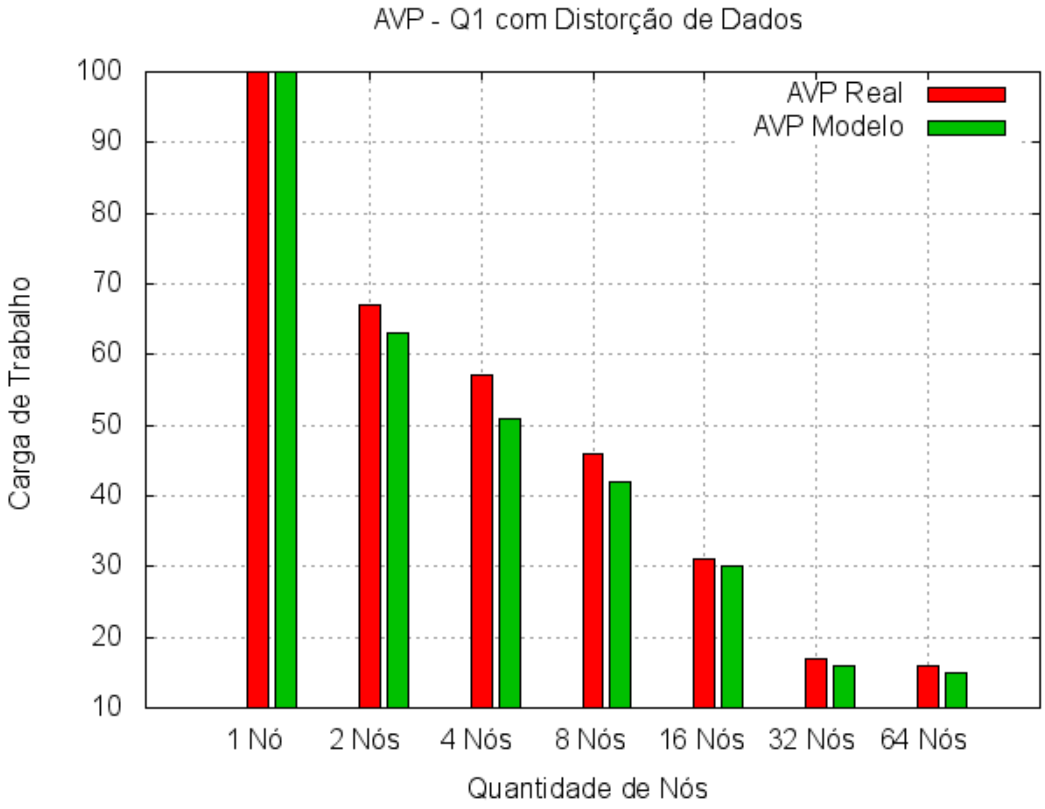

Figura 24. Consulta SQL com distorção de dados.

A figura 24 mostra a comparação dos resultados obtidos a partir do modelo e do ambiente real de uma consulta SQL com fragmentação AVP executada em um banco de dados com carga de trabalho desbalanceada em *clusters* com 2, 4, 8, 16, 32 e 64 nós. Foi verificado que o tempo de execução do modelo é inferior ao do ambiente real com variação entre 0,41% (32 nós) e 5,38% (4 nós) para todas as configurações de *cluster* utilizadas. Assim como no modelo SVP, a diferença de tempo verificada entre os dois ambientes pode ser justificada pelo fato da definição da carga de trabalho usada no modelo ter sido obtida a partir do gráfico de distorção de dados apresentado em [3].

4.4.3 Modelo AVP_WR

O modelo AVP_WR opera de maneira semelhante ao SVP e ao AVP, porém com a funcionalidade adicional de redistribuir dinamicamente a carga de trabalho entre os nós do *cluster* quando esta estiver desbalanceada. Quando um nó escravo termina de processar a carga de trabalho recebida do nó mestre ele propaga uma mensagem de oferta de ajuda para o nó escravo seguinte ativo no *cluster*, ou seja, o nó escravo 1

propaga para o nó escravo 2, o nó 2 para o nó 3, e assim sucessivamente até o último nó ativo, que propaga para o nó 1 quando termina de processar sua carga. Se primeiro nó a receber a mensagem de ajuda ainda possuir carga a ser processada ele aceita a oferta e imediatamente repassa metade da carga ao nó ofertante. Por exemplo, suponha que o nó 1 terminou de processar sua carga de trabalho e ofertou ajuda ao nó 2, que ainda possui carga de trabalho **W** = 4 pendente de processamento. O nó 2 então repassa carga de trabalho igual a **W** = 2 para o nó 1 e continua a processar o restante. Caso a carga de trabalho pendente seja **W** = 5, o nó que aceitou a ajuda repassa ao nó ofertante uma carga de trabalho **W** = (5 - 1) / 2, ou seja, **W** = 2. Caso o nó escravo seguinte não esteja necessitando de ajuda no momento, ele repassa a mensagem de oferta para o próximo nó e assim sucessivamente até que esta alcance todos os nós escravos do *cluster*.

Quando a carga de trabalho está totalmente balanceada o AVP_WR trabalha da mesma maneira que o AVP. Por esta razão esta lógica de operação não será discutida nesta seção. O algoritmo do modelo AVP_WR está descrito no anexo VIII deste trabalho.

A figura 25 apresenta o diagrama de transição de estados do modelo SVP com apenas dois nós escravos e carga de trabalho igual a seis. Neste exemplo, o estado 1 caracteriza o período de inatividade, enquanto os estados de 2 a 19 indicam que os nós escravos estão processando carga de trabalho demandada pelo nó mestre.

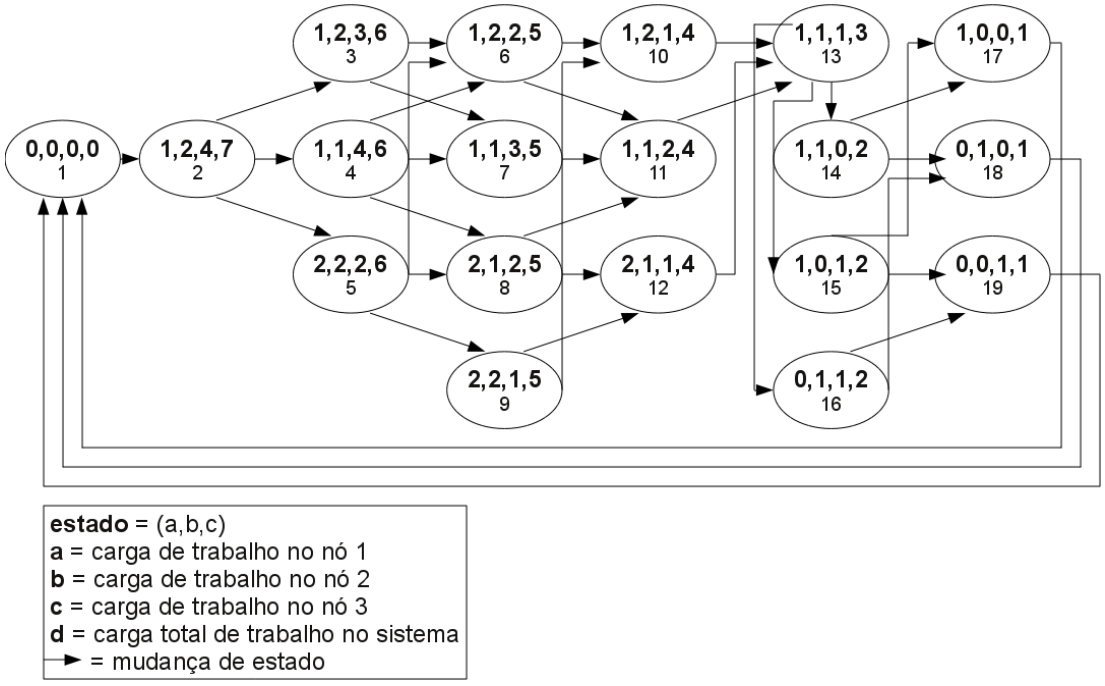

Figura 25. Modelo AVP_WR com 3 nós escravos e carga de trabalho igual a 7.

Assim como os modelos SVP e AVP, o AVP_WR também tem matriz de taxas de transição com estrutura quase diagonal. A figura 26, mostra matriz de transição do modelo apresentado na figura 25.

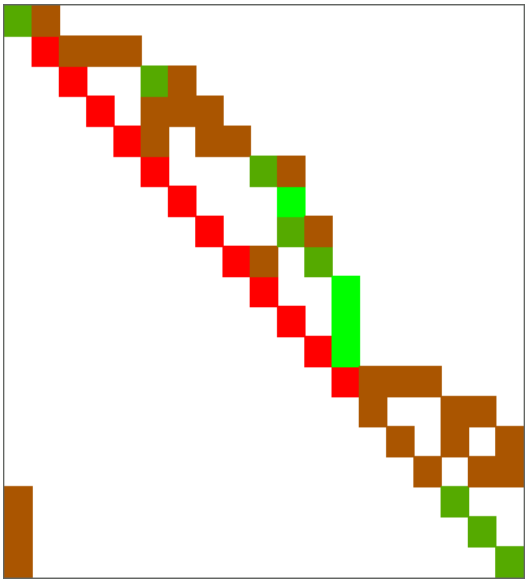

Figura 26. Matriz de taxas de transição com estrutura quase diagonal.

4.4.3.1 Comparação de Resultados entre AVP_WR Real e AVP_WR Modelado

Assim como nos modelos SVP e AVP_WR, foi realizada a comparação dos resultados obtidos por [3] com os resultados da modelagem analítica do modelo AVP_WR para *clusters* com 2, 4, 8, 16, 32 e 64. As consultas utilizadas também foram as mesmas, ou seja, SQL Q21, com carga de trabalho balanceada, e Q1, com carga desbalanceada, cuja distribuição de carga é a mesma apresentada na figura 20.

As figuras 27 e 28 mostram os gráficos com os tempos de execução normalizados em 100 obtidos por [3] e da simulações sobre o modelo AVP_WR realizadas nesta pesquisa.

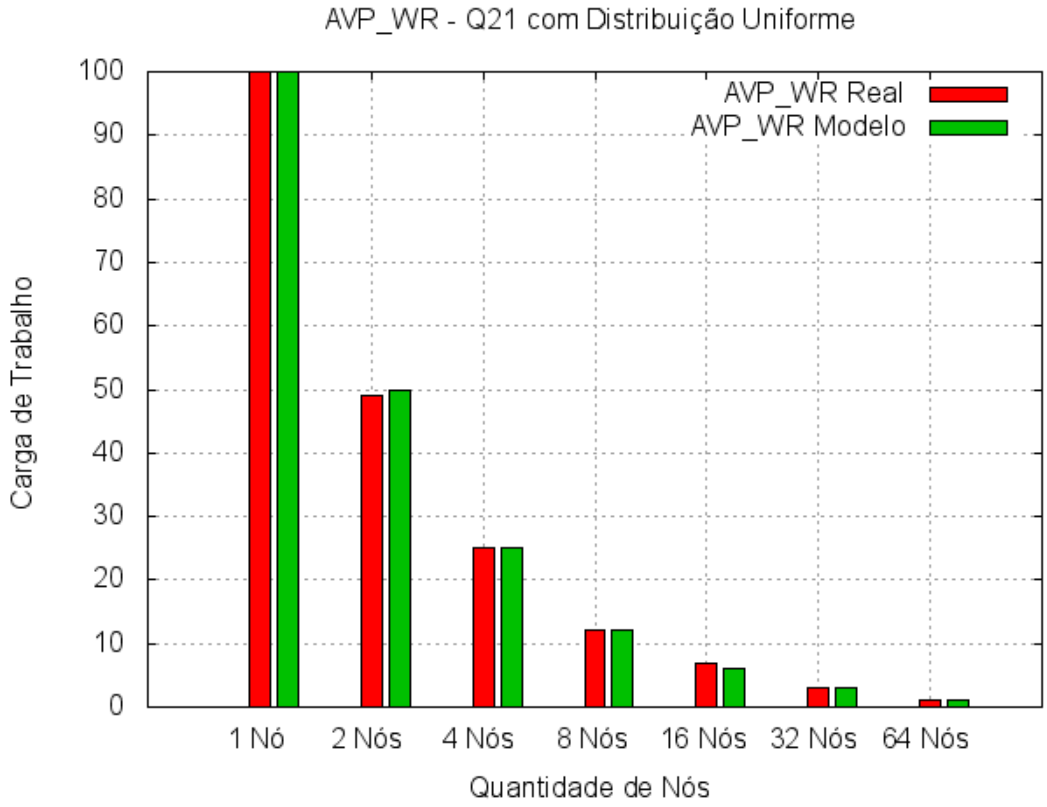

Figura 27. Consulta SQL com distribuição uniforme de dados.

A figura 27 mostra que os resultados obtidos a partir do modelo AVP_WR para uma consulta SQL executada em um banco de dados uniformemente distribuído são bastante semelhantes aos obtidos nos testes empíricos realizados em [3]. A configuração de *cluster* com 2 nós no ambiente modelado apresentou acréscimo de 0,64% de tempo em relação ao teste realizado no ambiente real AVP_WR. Em todas as demais configurações de *cluster* avaliadas o ambiente real AVP_WR apresentou ligeiro acréscimo de tempo em relação ao modelo. As diferenças observadas foram 0,46% (4 nós), 0,02% (8 nós), 0,82% (16 nós), 0,40% (32 nós) e 0,28% (64 nós).

A figura 28 exibe a comparação dos resultados obtidos a partir do modelo e do ambiente real de uma consulta SQL com fragmentação AVP_WR executada em um banco de dados com carga de trabalho desbalanceada em *clusters* com 2, 4, 8, 16, 32 e 64 nós. Foi verificado que o tempo de execução do modelo é ligeiramente superior ao do ambiente real em todas as configurações de *cluster* avaliadas. As diferenças observadas foram as seguintes: 3,2% para 2 nós; 0,75% para 4 nós; 0,63% para 8 nós; 0,45% para 16 nós; 0,26% para 32 nós; e 0,07% para 64 nós.

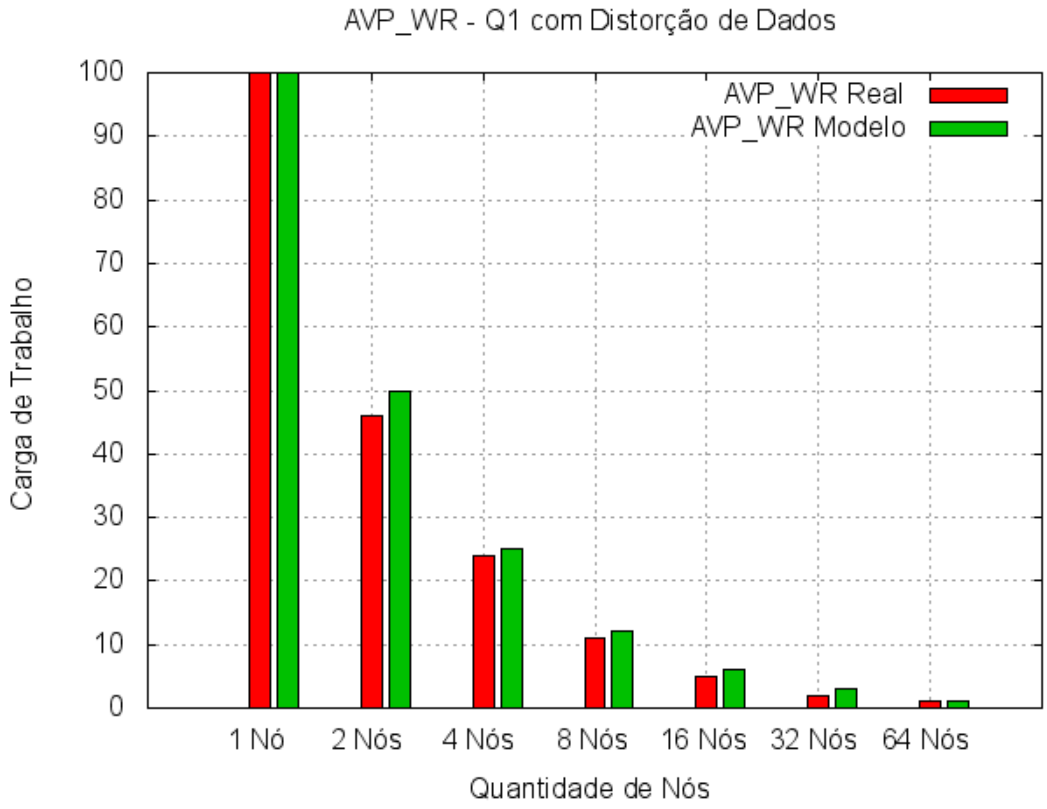

Figura 28. Consulta SQL com distorção de dados.

4.4.4 Avaliação dos Modelos

A partir dos resultados obtidos com os três modelos estudados é possível concluir que o grau de abstração aplicado nos mesmos é suficiente para uma boa representação do comportamento das técnicas de fragmentação virtual de dados SVP, AVP e AVP_WR. Foram realizados seis experimentos, sendo dois para cada um dos modelos, um com distribuição uniforme e outro com distorção de dados. Após a normalização dos resultados foi verificado que os tempos de execução apresentaram pequena variação entre os testes realizados nos ambientes real e modelado. A maior diferença ocorreu na consulta Q1, com distorção de dados, realizada no modelo SVP, onde foi observada uma diferença de 12,34% em relação ao ambiente real. Essa diferença pode ser justificada pelo fato da carga de trabalho utilizada ter sido obtida a partir do gráfico de distorção de dados apresentado em [3] e não por informações baseada em valores explícitos. A menor diferença ocorreu na consulta Q21, com

distribuição uniforme, realizada no modelo AVP_WR, onde foi verificada uma diferença de apenas 0,02% em relação ao ambiente real.

A partir dos modelos estudados é possível concluir que:

- Não há diferença de tempo total de execução das consultas quando estas são realizadas em apenas um nó, pois nesta situação não há divisão de carga de trabalho;
- O SVP não realiza redistribuição de carga, por isso o tempo de execução de uma consulta que utiliza esta técnica é sempre determinado pelo nó escravo responsável por processar a maior carga de trabalho, desde que todos os nós do *cluster* possuam a mesma capacidade de processamento;
- O AVP executa a divisão das consultas em fragmentos menores, característica representada pela taxa de serviço no respectivo modelo, e por esta razão, caso ainda assim não seja possível dividi-la com justiça entre os nós escravos, o tempo de execução de uma consulta também será determinado pelo nó responsável por processar a maior carga de trabalho. Caso a carga seja dividida igualmente entre os nós, os mesmos concluem suas respectivas tarefas ao mesmo tempo;
- O AVP_WR, assim como o AVP, efetua a divisão das consultas em fragmentos menores, porém, caso esta divisão não seja realizada com justiça ou a capacidade de processamento de algum nó seja superior aos demais, o algoritmo ajusta dinamicamente durante o processamento da consulta a divisão da carga de trabalho entre os nós. Desta maneira, todos os nós concluem suas tarefas quase simultaneamente, pois há uma pequena fração adicional de processamento gerada em função da execução do algoritmo de redistribuição dinâmica da carga de trabalho.

Estas mesmas conclusões foram obtidas por [3] no ambiente real.

5 Conclusão

Este trabalho apresentou o desenvolvimento de um *software* livre, denominado UAS (UNIRIO Analisador de Sistemas), cujo objetivo é prover o monitoramento de recursos de ambientes distribuídos através de uma interface de gerenciamento amigável que consome poucos recursos computacionais e oferece facilidade de instalação e configuração. Além disso, a ferramenta fornece uma funcionalidade não encontrada na literatura, que é a conversão automática do arquivo de saída do utilitário TCPDump em um arquivo de entrada para a ferramenta de simulação de rede *Network Simulator* 2. Este módulo da ferramenta foi denominado "TCPDump2NS".

Embora existam diversas ferramentas livres para o monitoramento de ambientes de rede que ofereçam robustez, precisão, grande variedade de opções de parametrização e análise de desempenho, poucas apresentam facilidade de configuração, sobretudo no que se refere à inclusão de clientes, e nenhuma oferece o recurso de replicação imediata de um ambiente real em ambiente simulado. Esta funcionalidade pode auxiliar pesquisadores e analistas na tarefa de identificação de uma configuração de rede ideal para um determinado sistema.

A UAS foi utilizada no laboratório do projeto CG-OLAP, pesquisa desenvolvida pela UNIRIO financiada pela FAPERJ. Este laboratório consiste em um *cluster* de banco de dados geográfico que utiliza fragmentação virtual de dados. Em um dos testes a ferramenta monitorou e coletou informações do tráfego de dados no ambiente durante 16,8 minutos, tempo de duração do processamento de um conjunto de consultadas definidas no *Star Schema Benchmark* [20]. Em seguida, o arquivo de *log* foi convertido pelo TCPDump2NS para então serem realizadas a validação desta funcionalidade e os testes de simulação de alteração de configuração do ambiente.

O primeiro teste de simulação promoveu a alteração de uma série de parâmetros de rede. Foram combinadas e simuladas doze configurações diferentes de parâmetros de rede. Os resultados indicaram que em razão de suas características, o *cluster* CG-OLAP não sofrerá alterações significativas de desempenho ainda que qualquer das alterações promovidas no ambiente simulado seja implementada no ambiente real.

O segundo teste de simulação consistiu na projeção de comportamento do ambiente CG-OLAP em eventos de inclusão e remoção de nós do *cluster*. Nestes testes foi executado o mesmo conjunto de consultas do teste anterior. O resultado obtido a partir da simulação da inclusão de um nó no *cluster* indicou uma diferença de 3,24% (19,9 segundos) quando comparado com o ambiente real na mesma situação. Já o evento de remoção de um nó apontou uma diferença de 6,79% (67,3 segundos) em comparação com o ambiente real. Estes valores se encontram dentro da variância, que gira em torno de 10%. Estes resultados mostram que o ambiente de simulação é coerente com o ambiente real, e que esta é uma funcionalidade bastante útil nos estudos de prevenção de problemas ou planejamento de crescimento de sistemas.

A outra proposta desta dissertação foi a modelagem markoviana de consultas SQL com fragmentação virtual de dados em *clusters* de banco de dados. A criação de um modelo matemático tem como base valores conhecidos ou previstos pelo sistema real, e são avaliados através da comparação com as medidas obtidas no ambiente real. O uso de modelos matemáticos possibilita a redução de execução de experimentos empíricos no ambiente real, uma vez que o modelo experimental pode fornecer respostas à questões particulares baseadas em hipóteses.

Foram desenvolvidos modelos matemáticos markovianos para três técnicas de fragmentação virtual de dados: Particionamento Virtual Simples (*Simple Virtual Partitioning* – SVP); Particionamento Virtual Adaptativo (*Adaptive Virtual Partitioning* – AVP), e; Particionamento Virtual Adaptativo com Redistribuição Dinâmica de Carga **(***Adaptive Virtual Partitioning Workload Redistribution* – AVP_WR). Para validação foram comparados os resultados obtidos a partir dos modelos com os resultados obtidos por LIMA em [3], tendo como base os tempos de execução de duas consultas em configurações de *clusters* com 2, 4, 8, 16, 32 e 64 nós.

Os resultados obtidos a partir dos modelos desenvolvidos mostram que o grau de abstração aplicado é suficiente para representar o comportamento das técnicas de

fragmentação virtual de dados SVP, AVP e AVP_WR quando os nós do *cluster* forem idênticos. Foram realizados dois experimentos para cada um dos modelos, um com distribuição uniforme e outro com distorção de dados.

Foi verificado que os tempos de execução apresentaram pequena variação entre os ambientes real e modelado. A maior diferença foi observada no modelo SVP na consulta com distorção de dados, que apresentou diferença de 12,34% em relação ao ambiente real na configuração de *cluster* com 16 nós. Esta diferença pode ser justificada em razão da carga de trabalho utilizada ter sido obtida a partir do gráfico de distorção de dados apresentado em [3], e não por informações baseada em valores explícitos. A menor diferença foi observada no modelo AVP_WR, cuja consulta com distribuição uniforme apresentou uma diferença de apenas 0,02% em relação ao ambiente real na configuração de *cluster* com 8 nós. Desta maneira, é possível concluir que os três modelos desenvolvidos nesta pesquisa podem ser utilizados como instrumento de avaliação de desempenho de consultas SQL que utilizam as técnicas de fragmentação de dados SVP, AVP e AVP_WR.

Como trabalhos futuros, pretende-se aprimorar o módulo TCPDump2NS da ferramenta UAS para que testes de inclusão e remoção de nós da rede sejam ainda mais apurados e fáceis de serem realizados. Espera-se também realizar mais testes com as ferramentas UAS e NS-2 em outros ambientes acadêmicos onde de fato a alteração de parâmetros de rede influenciem diretamente no comportamento do ambiente. Verificar o comportamento do sistema em caso de falha de um ou mais nós e implementar um algoritmo que represente esta situação. Por fim, faz-se necessário implementar soluções para o problema das explosões de estados nos modelos markovianos desenvolvidos no TANGRAM-II.

6 Referências

[1] AKAL, F., BÖHM, K., AND SCHEK, H.-J., “OLAP Query Evaluation in a Database Cluster: a Performance Study on Intra-Query Parallelism”, In: *Proceedings of the East European Conf. on Advances in Databases and Information Systems (ADBIS), 6th European East Conference*, pp. 218-231, Bratislava, Slovakia, 2002.

[2] MATTOSO, M., ZIMBRÃO G., LIMA, A., BAIÃO, F. et al., “ParGRES: uma camada de processamento paralelo de consultas sobre o PostgreSQL”. In: *Workshop de Software Livre*, pp. 259-264, Porto Alegre, 2005.

[3] LIMA, A., “Paralelismo Intra-Consulta em Clusters de Bancos de Dados”, *Tese de Doutorado, Programa de Pós-Graduação de Engenharia de Sistemas e Computação*, COPPE, UFRJ, Brasil, 2004.

[4] ÖSZU, T., VALDURIEZ, P., *Principles of Distributed Database Systems*. 2a. ed. New Jersey, Prentice-Hall, 1999.

[5] Cacti (2009), “*The Complete RRDtool-based Graphing Solution*”, http://cacti.net. Acessado em 28/06/2009.

[6] NTOP (1998), “*Network TOP*”, http://www.ntop.org. Acessado em 30/06/2009.

[7] OMNet++ (1998), "*A Discrete Event Simulation Environment*", http://www.omnetpp.org. Acessado em 18/01/2010.

[8] NS-2 (1989), "*The Network Simulator*", http://isi.edu/nsnam/ns. Acessado em 12/01/2010.

[9] CASTRO, E. F. R., DINIZ, M. C., BAIÃO, F. A., "Uma Ferramenta Livre para Análise de Desempenho e Simulação de Ambientes Distribuídos". In: *25º Simpósio Brasileiro de Banco de Dados*, Belo Horizonte, 2010.

[10] ASTRAHAM, M. M., CHAMBERLIN D. D., "Implementation of a Structured English Query Language", In: *Communications of the ACM*, New York, v. 18, pp. 580-588, 1975.

[11] CARMO, R. M. L. R., CARVALHO, L. R., SILVA, E. S., DINIZ, M. C., MUNTZ, R. R., "Performance/Availability Modeling with the TANGRAM-II Modeling Environment", In: *Performance Evaluation*, v. 33, pp. 45-65, 1998.

[12] HEUSER, C. A., *Projeto de Banco de Dados*. 4a. ed. Porto Alegre, Sagra Luzzato, 1998.

[13] ELMASRI, R., NAVATHE, S. B., *Sistemas de Banco de Dados*. 4a. ed. São Paulo, Pearson Addison Wesley, 2008.

[14] STERLING, T., "An Introduction to PC Clusters for High Performance Computing", In: *The International Journal of High Performance Computing Applications*, v. 15, n. 2, pp. 92-101, 2001.

[15] RIGAUX, P., SCHOLL, M., VOISARD, A., *Spatial Databases with application to GIS*. San Francisco, Morgan Kaufmann, 2002.

[16] WALTON, C. B., DALE, A. G., JENEVEIN, R. M., "A Taxonomy and Performance Model of Data Skew Effects in Parallel Joins", In: *Proceedings of the*

17th International Conference on Very Large Databases (VLDB), pp. 537–548, Barcelona, 1991.

[17] GORLA, N., "Features to Consider in a Data Warehousing System", In: *Communications of the ACM*, New York, v. 46, pp. 111-115, 2003.

[18] MATTOSO, M., ZIMBRÃO G., LIMA, A., BAIÃO, F. et al., "ParGRES: Middleware para Processamento Paralelo de Consultas OLAP em Clusters de Banco de Dados". In: *20º Simpósio Brasileiro de Banco de Dados*, Uberlândia, 2005.

[19] PostgreSQL (2009), "*Open Source Object-Relational Database System*", http://www.postgresql.org. Acessado em 20/01/2010.

[20] O'NEIL, P., O'NEIL, B., CHEN, X. (2007), "*The Star Schema Benchmark (SSB)*", http://www.cs.umb.edu/~xuedchen/research/publications/StarSchemaB.PDF. Acessado em 11/01/2010.

[21] PHP (2009), "*A Widely-Used General-Purpose Scripting Language*", www.php.net. Acessado em 09/06/2009.

[22] Bash Shell (1978), "*Bourne-Again Shell - A Command Interpreter and a Programming Language*", http://www.gnu.org/software/bash. Acessado em 20/01/2010.

[23] PHP SysInfo (2009), "*A PHP Script that Displays Information About the Host Being Accessed*", http://phpsysinfo.sourceforge.net. Acessado em 09/06/2009.

[24] GNU General Public License (2007), "*GNU Operating System*", http://www.gnu.org/copyleft/gpl.html. Acessado em 20/01/2010.

[25] Apache (2009), "*The Apache HTTP Server*", http://httpd.apache.org. Acessado em 09/06/2009.

[26] NET-SNMP (2008), "*Simple Network Management Protocol*", http://www.net-snmp.org. Acessado em 13/07/2009.

[27] MySQL (2007) ,"*An Open Source Database Software*", http://www.mysql.com. Acessado em 28/06/2009.

[28] RRDTool (2007), "*A High Performance Data Logging and Graphing System for Time Series Data*", http://oss.oetiker.ch/rrdtool. Acessado em 28/06/2009.

[29] MRTG (2007), "*The Multi Router Traffic Grapher*", http://oss.oetiker.ch/mrtg. Acessado em 12/07/2009.

[30] ROCHOL, J., SOUZA, L. D., SEWALD, L., FERNANDES, R. H., MORI, O. N. (2003), "*Plataformas de Simulação de Software Livre para Redes Fixas e Móveis: Características, Suporte, Instalação e Validação*", http://nsl.csie.nctu.edu.tw/NCTUnsReferences/paper76-I2TS2003.pdf. Acessado em 02/04/2010.

[31] RÖHM, U., BÖHM, K., SCHEK, H. J., SCHULDT, H., "FAS - A Freshness-Sensitive Coordination Middleware for a Cluster of OLAP Components". In: *Proceedings of the 28th International Conference on Very Large Data Bases*, Hong Kong, pp. 754-765, 2002.

[32] SILVA, E. S., MUNTZ, R. R., "Métodos Computacionais de Solução de Cadeias de Markov: Aplicações a Sistemas de Computação e Comunicação". In: *VIII Escola de Computação*, Gramado, pp. 187-195, 1992.

[33] AJMONE-MARSAN, M., BALBO, G, CONTE, G., "Performance Models of Multiprocessor Systems". In: *Cambridge: MIT Press Series in Computer Systems*, 1986.

[34] MARKOV, A. ª, “Extension of the limit theorems of probability theory to a sum of variables connected in a chain”, reprinted in Appendix B of: R. HOWARD. *Dynamic Probabilistic Systems, volume 1: Markov Chains*. John Wiley and Sons, 1971.

[35] TRIVEDI, K. S., *Probability and Statistics with Reliability, Queuing and Computer Science Applications*. 2a. ed. New York, John Wiley and Sons, 2001.

[36] KLEINROCK, L., *Queueing Systems - Theory and Practice*. New York, Wiley – Interscience Publication. v.1, 1975.

[37] SILVA, E. S., GAIL, H. R., “Transient Solution for Markov Chains”, In: *Computational Probability*, pp. 44-79, Kluwer, 2000.

[38] GOYAL, A. CARTER, W. C., SILVA, E. S., LAVENBERG, S. S., TRIVEDI, K. H., “The system Availability Estimator”, In: *Proceedings of the 6*th *Annual International Symposium on Fault-Tolerant Computing Systems (FCTS-16)*, Viena, pp. 84-89, 1986.

[39] SILVA, E. S., LEÃO R. M. M., DUARTE, F. P. et al., “Modelagem e Análise de Redes com o Conjunto de Ferramentas TANGRAM-II”. In: *Simpósio Brasileiro de Redes de Computadores*, pp. 897-904, Curitiba, 2006.

[40] MEO, M., SILVA, E. S., MARSAN, M. A., “Efficient Solution for a Class of Markov Chain Models of Telecommunication Systems”, In: *Performance Evaluation*, Amsterdam, v. 27-28, pp. 603-625, 1996.

7 Anexos

Anexo I - Consultas Multidimensionais

a) Qual a receita gerada no ano de 1993 pelas vendas onde foi aplicado o desconto na faixa de 1 a 3 e a quantidade de itens vendidos foi menor que 25:
*SELECT sum(lo_extendedprice*lo_discount) as revenue FROM dwg.f_lineorder f, dwg.d_time t WHERE f.sky_time_orderdate = t.sky_time and d_year = 1993 and lo_discount between 0.01 and 0.03 and lo_quantity < 25;*

b) Qual a receita gerada no mês de janeiro do ano de 1994 pelas vendas onde foi aplicado o desconto na faixa de 4 a 6 e a quantidade de itens vendidos ficou entre 26 e 35 unidades:
*SELECT sum(lo_extendedprice*lo_discount) as revenue FROM dwg.f_lineorder f, dwg.d_time t WHERE f.sky_time_orderdate = t.sky_time and d_year = 1994 and d_month = 01 and lo_discount between 0.04 and 0.1 and lo_quantity between 26 and 150;*

c) Qual a receita gerada na sexta semana do ano de 1994 pelas vendas onde foi aplicado o desconto na faixa de 5 a 7 e a quantidade de itens vendidos ficou entre 26 e 35 unidades:
*SELECT sum(lo_extendedprice*lo_discount) as revenue FROM dwg.f_lineorder f, dwg.d_time t WHERE f.sky_time_orderdate = t.sky_time and d_weekofyear = 6 and d_year = 1994 and lo_discount between 5 and 7 and lo_quantity between 26 and 35;*

d) Qual a receita gerada por ano e tipo de peça para peças da categoria 'MFGR#12' cujo fornecedor esteja na região norte:
*SELECT sum(lo_extendedprice*lo_discount) as revenue, d_year, p_type FROM dwg.f_lineorder f, dwg.d_time t, dwg.d_part p, dwg.d_supplier s, dwg.d_geometry sg WHERE f.sky_time_orderdate = t.sky_time and p.sky_part = f.sky_part and s.sky_supplier = f.sky_supplier and s.sky_geometry = sg.sky_geometry and p_category = 'MFGR#12' and sg.g_region = 'Norte' group by d_year, p_type order by d_year, p_type;*

e) Qual a receita gerada por ano e tipo de peça para peças cujo tipo esteja entre 'MFGR#2221' e 'MFGR#2228' e o fornecedor esteja na região nordeste:
*SELECT sum(lo_extendedprice*lo_discount) as revenue, d_year, p_type FROM dwg.f_lineorder f, dwg.d_time t, dwg.d_part p, dwg.d_supplier s, dwg.d_geometry sg WHERE f.sky_time_orderdate = t.sky_time and p.sky_part = f.sky_part and s.sky_supplier = f.sky_supplier and s.sky_geometry = sg.sky_geometry and p_type between 'MFGR#2221' and 'MFGR#2228' and sg.g_region = 'Nordeste' group by d_year, p_type order by d_year, p_type;*

f) Qual a receita gerada por ano e tipo de peça para peças do tipo 'MFGR#2239' cujo fornecedor esteja na região Sul:
*SELECT sum(lo_extendedprice*lo_discount) as revenue, d_year, p_type FROM wg.f_lineorder f, dwg.d_time t, dwg.d_part p, dwg.d_supplier s, dwg.d_geometry sg WHERE f.sky_time_orderdate = t.sky_time and p.sky_part = f.sky_part and s.sky_supplier = f.sky_supplier and s.sky_geometry = sg.sky_geometry and p_type= 'MFGR#2239' and sg.g_region = 'Sul' group by d_year, p_type order by d_year, p_type;*

g) Qual a receita gerada por ano, agrupada pelos estados dos compradores e dos fornecedores, onde ambos pertençam a região “Sudeste” ocorridas entre 1992 e 1997:
*SELECT cg.g_state, sg.g_state, d_year, sum(lo_extendedprice*lo_discount) as revenue FROM dwg.d_customer c, dwg.f_lineorder f, dwg.d_supplier s, dwg.d_time t, dwg.d_geometry cg, dwg.d_geometry sg WHERE f.sky_customer = c.sky_customer*

and f.sky_supplier = s.sky_supplier and f.sky_time_orderdate = t.sky_time and c.sky_geometry = cg.sky_geometry and s.sky_geometry = sg.sky_geometry and cg.g_region = 'Sudeste' and sg.g_region = 'Sudeste' and d_year >= 1992 and d_year <= 1997 group by cg.g_state, sg.g_state, d_year order by d_year asc, revenue desc;

h) Qual a receita gerada por ano, agrupada pelas cidades dos compradores e dos fornecedores, onde ambos pertençam ao "Rio de Janeiro", ocorridas entre 1992 e 1997:

*SELECT cg.g_city, sg.g_city, d_year, sum(lo_extendedprice*lo_discount) as revenue FROM dwg.d_customer c, dwg.f_lineorder f, dwg.d_supplier s, dwg.d_time t,dwg.d_geometry cg, dwg.d_geometry sg WHERE f.sky_customer = c.sky_customer and f.sky_supp = s.sky_supplier and f.sky_time_orderdate = t.sky_time and c.sky_geometry = cg.sky_geometry and s.sky_geometry = sg.sky_geometry and cg.g_state = 'Rio de Janeiro' and sg.g_state = 'Rio de Janeiro' and d_year >= 1992 and d_year <= 1997 group by cg.g_city, sg.g_city, d_year order by d_year asc, revenue desc;*

i) Qual a receita gerada por ano, agrupada pelas cidades dos compradores e dos fornecedores, onde ambos pertençam a uma das seguintes cidades "Niteroi" ou "Saquarema", ocorridas entre 1992 e 1997:

*SELECT cg.g_city, sg.g_city, d_year, sum(lo_extendedprice*lo_discount) as revenue FROM dwg.d_customer c, dwg.f_lineorder f, dwg.d_supplier s, dwg.d_time t,dwg.d_geometry cg, dwg.d_geometry sg WHERE f.sky_customer = c.sky_customer and f.sky_supplier = s.sky_supplier and f.sky_time_orderdate = t.sky_time And c.sky_geometry = cg.sky_geometry and s.sky_geometry = sg.sky_geometry and (cg.g_city='Niteroi' or cg.g_city='Saquarema') and (sg.g_city='Niteroi' or sg.g_city='Saquarema') and d_year >= 1992 and d_year <= 1997 group by cg.g_city, sg.g_city, d_year order by d_year asc, revenue desc;*

j) Qual a receita gerada por ano, agrupada pelas cidades dos compradores e dos fornecedores, onde ambos pertençam a uma das seguintes cidades "Niteroi" ou "Saquarema", ocorridas em Dezembro de 1997:

*SELECT cg.g_city, sg.g_city, d_year, sum(lo_extendedprice*lo_discount) as revenue FROM dwg.d_customer c, dwg.f_lineorder f, dwg.d_supplier s, dwg.d_time t, dwg.d_geometry cg, dwg.d_geometry sg WHERE f.sky_customer = c.sky_customer and f.sky_supplier = s.sky_supplier and f.sky_time_orderdate = t.sky_time And c.sky_geometry = cg.sky_geometry and s.sky_geometry = sg.sky_geometry and (cg.g_city='Niteroi' or cg.g_city='Saquarema') and (sg.g_city='Niteroi' or sg.g_city='Saquarema') and d_year = 1997 and d_month = 12 group by cg.g_city, sg.g_city, d_year order by d_year asc, revenue desc;*

k) Qual a receita líquida gerada por ano e estado do consumidor, onde a região de origem do consumidor e do fornecedor seja "Norte" e o fabricante das peças seja "MFGR#1" ou "MFGR#2":

*SELECT d_year, cg.g_state, sum(lo_extendedprice*lo_discount - lo_supplycost) as profit FROM dwg.d_customer c, dwg.f_lineorder f, dwg.d_supplier s, dwg.d_time t, dwg.d_part p, dwg.d_geometry cg, dwg.d_geometry sg WHERE f.sky_customer = c.sky_customer and f.sky_supplier = s.sky_supplier and f.sky_part = p.sky_part and f.sky_time_orderdate = t.sky_time And c.sky_geometry = cg.sky_geometry and s.sky_geometry = sg.sky_geometry and cg.g_region = 'Norte' and sg.g_region = 'Norte' and (p_mfgr = 'MFGR#1' or p_mfgr = 'MFGR#2') group by d_year, cg.g_state order by d_year, cg.g_state;*

l) Qual a receita líquida gerada por ano, estado do fornecedor e categoria de peças, onde a região de origem do consumidor e do fornecedor seja "Norte", o fabricante das peças seja "MFGR#1" ou "MFGR#2" e o ano seja 1997 ou 1998:

*SELECT d_year, sg.g_state, p_category, sum(lo_extendedprice*lo_discount - lo_supplycost) as profit FROM dwg.d_customer c, dwg.f_lineorder f, dwg.d_supplier s, dwg.d_time t, dwg.d_part p, dwg.d_geometry cg, dwg.d_geometry sg WHERE f.sky_customer = c.sky_customer and f.sky_supp = s.sky_supplier and f.sky_part = p.sky_part and f.sky_time_orderdate = t.sky_time And c.sky_geometry = cg.sky_geometry and s.sky_geometry = sg.sky_geometry and cg.g_region = 'Norte' and sg.g_region = 'Norte' and (d_year = 1997 or d_year = 1998) and (p_mfgr =*

'MFGR#1' or p_mfgr = 'MFGR#2') group by d_year, sg.g_state, p_category order by d_year, sg.g_state, p_category;

m) Qual a receita líquida gerada por ano, cidade do fornecedor e tipo de peças, onde o estado do fornecedor seja “Amazonas”, a categoria das peças seja “MFGR#14” e o ano seja 1997 ou 1998:

*SELECT d_year, sg.g_city, p_brand1, sum(lo_extendedprice*lo_discount - lo_supplycost) as profit FROM dwg.d_customer c, dwg.f_lineorder f, dwg.d_supplier s, dwg.d_time t, dwg.d_part p, dwg.d_geometry sg WHERE f.sky_customer = c.sky_customer and f.sky_supplier = s.sky_supplier and f.sky_part = p.sky_part and f.sky_time_orderdate = t.sky_time and s.sky_geometry = sg.sky_geometry and sg.g_state = 'Amazonas' and (d_year = 1997 or d_year = 1998) and p_category = 'MFGR#14' group by d_year, sg.g_city, p_brand1 order by d_year, sg.g_city, p_brand1;*

Anexo II - Consultas Multidimensionais Geográficas

a) Qual a receita gerada no ano de 1993 pelas vendas onde foi aplicado o desconto na faixa de 1 a 3 a quantidade de itens vendidos foi menor que 25 e a cidade do cliente tem menos que 5000 unidades de área:

*SELECT sum(lo_extendedprice*lo_discount) as revenue FROM dwg.f_lineorder f, dwg.d_time t, dwg.d_customer c, dwg.d_geometry g WHERE f.sky_time_orderdate = t.sky_time and f.sky_customer = c.sky_customer and c.sky_geometry = g.sky_geometry and d_year = 1993 and lo_discount between 0.01 and 0.03 and lo_quantity < 25 and area(g_map) < 5000;*

b) Qual a receita gerada entre janeiro e junho do ano de 1994 pelas vendas onde foi aplicado o desconto na faixa de 4 a 10, a quantidade de itens vendidos ficou entre 26 e 150 unidades e a cidade do fornecedor tem mais que 500 unidades de área:

*SELECT sum(lo_extendedprice*lo_discount) as revenue FROM dwg.f_lineorder f, dwg.d_time t, dwg.d_supplier s, dwg.d_geometry g WHERE f.sky_time_orderdate = t.sky_time and f.sky_supplier = s.sky_supplier and s.sky_geometry = g.sky_geometry and d_year = 1994 and d_month between 01 and 06 and lo_discount between 0.04 and 0.1 and lo_quantity between 26 and 150 and area(g_map) > 500;*

c) Qual a receita gerada na sexta semana do ano de 1994 pelas vendas onde foi aplicado o desconto na faixa de 5 a 7 (5 a 7% -??), a quantidade de itens vendidos ficou entre 26 e 35 unidades e a distância entre a cidade do cliente e a do fornecedor seja menor que 1000000 unidades:

*SELECT sum(lo_extendedprice*lo_discount) as revenue FROM dwg.f_lineorder f, dwg.d_time t, dwg.d_supplier s, dwg.d_geometry sg, dwg.d_customer c, dwg.d_geometry cg WHERE f.sky_time_orderdate = t.sky_time and f.sky_supplier = s.sky_supplier and s.sky_geometry = sg.sky_geometry and f.sky_customer = c.sky_customer and c.sky_geometry = cg.sky_geometry and d_weekofyear = 6 and d_year = 1994 and lo_discount between 5 and 7 and lo_quantity between 26 and 35 and distance(cg.g_map, sg.g_map) < 1000000;*

d) Qual a receita gerada por ano, tipo de peça, cidade e país do cliente para peças da categoria 'MFGR#12' cujo fornecedor esteja na região Norte o e a cidade do cliente seja vizinha a cidade do fornecedor:

*SELECT sum(lo_extendedprice*lo_discount) as revenue, d_year, p_type, cg.g_city, cg.g_nation FROM dwg.f_lineorder f, dwg.d_time t, dwg.d_part p, dwg.d_supplier s, dwg.d_geometry sg, dwg.d_customer c, dwg.d_geometry cg WHERE f.sky_time_orderdate = t.sky_time and f.sky_part = p.sky_part and f.sky_supplier = s.sky_supplier and s.sky_eometry = sg.sky_geometry and f.sky_customer = c.sky_customer and c.sky_geometry = cg.sky_geometry and p_category = 'MFGR#12' and sg.g_region = 'Norte' and equals (cg.g_map, sg.g_map) = 't' group by d_year, p_type, cg.g_city, cg.g_nation order by d_year, p_type, cg.g_city, cg.g_nation;*

e) Qual a receita gerada por ano e tipo de peça para peças cujo tipo esteja entre 'MFGR#2221' e 'MFGR#2228' , o fornecedor esteja na região Nordeste e a distância entre a cidade do cliente e a do fornecedor esteja entre 500000 e 1200000 unidades, inclusive:

*SELECT sum(lo_extendedprice*lo_discount) as revenue, d_year, p_type FROM dwg.f_lineorder f, dwg.d_time t, dwg.d_part p, dwg.d_supplier s, dwg.d_geometry sg, dwg.d_customer c, dwg.d_geometry cg WHERE f.sky_time_orderdate = t.sky_time and f.sky_part = p.sky_part and f.sky_supplier = s.sky_supplier and s.sky_geometry = sg.sky_geometry and f.sky_customer = c.sky_customer and c.sky_geometry = cg.sky_geometry and p_type between 'MFGR#2221' and 'MFGR#2228' and sg.g_region = 'Nordeste' and distance(cg.g_map, sg.g_map) >= 500000 and distance(cg.g_map, sg.g_map) <= 1200000 group by d_year, p_type order by d_year, p_type;*

f) Qual a receita gerada por ano, tipo de peça, cidade e país do fornecedor para peças do tipo 'MFGR#2239' cujo fornecedor esteja na região Sul e a cidade do cliente não seja vizinha a cidade do fornecedor:

*SELECT sum(lo_extendedprice*lo_discount) as revenue, d_year, p_type, sg.g_city, sg.g_nation FROM dwg.f_lineorder f, dwg.d_time t, dwg.d_part p, dwg.d_supplier s, dwg.d_geometry sg, dwg.d_customer c, dwg.d_geometry cg WHERE*

f.sky_time_orderdate = t.sky_time and f.sky_part = p.sky_part and f.sky_supplier = s.sky_supplier and s.sky_geometry = sg.sky_geometry and f.sky_customer = c.sky_customer and c.sky_geometry = cg.sky_geometry and p_type= 'MFGR#2239' and sg.g_region = 'Sul' and touches (cg.g_map, sg.g_map) = 'f' group by d_year, p_type, sg.g_city, sg.g_nation order by d_year, p_type, sg.g_city, sg.g_nation;

g) Qual a receita gerada por ano ocorridas entre 1992 e 1997, agrupada pelos países dos compradores e dos fornecedores, onde ambos pertençam a mesma cidade:
*SELECT cg.g_nation as C_Nation, sg.g_nation as S_Nation, d_year, sum(lo_extendedprice*lo_discount) as revenue FROM dwg.d_customer c, dwg.f_lineorder f, dwg.d_supplier s, dwg.d_time t, dwg.d_geometry cg, dwg.d_geometry sg WHERE f.sky_customer = c.sky_customer and f.sky_supplier = s.sky_supplier and f.sky_time_orderdate = t.sky_time and c.sky_geometry = cg.sky_geometry and s.sky_geometry = cg.sky_geometry and equals (cg.g_map,sg.g_map)='t' and d_year >= 1992 and d_year <= 1997 group by C_Nation, S_Nation, d_year order by C_Nation, _Nation, d_year asc, revenue desc;*

h) Qual a receita gerada por ano, agrupada pelas cidades dos compradores e dos fornecedores, onde suas cidades façam fronteira, ocorridas entre 1992 e 1997.
*SELECT cg.g_city as C_City, sg.g_city as S_City, d_year, sum(lo_extendedprice*lo_discount) as revenue FROM dwg.d_customer c, dwg.f_lineorder f, dwg.d_supplier s, dwg.d_time t, dwg.d_geometry cg, dwg.d_geometry sg WHERE f.sky_customer = c.sky_customer and f.sky_supplier = s.sky_supplier and f.sky_time_orderdate = t.sky_time and c.sky_geometry = cg.sky_geometry and s.sky_geometry = cg.sky_geometry and touches(cg.g_map,sg.g_map)='t' and d_year >= 1992 and d_year <= 1997 group by C_City, S_City, d_year order by d_year asc, revenue desc;*

i) Qual a receita gerada por ano, agrupada pelas cidades dos compradores e dos fornecedores, onde a cidade do fornecedor não faça fronteira com a cidade do comprador, ocorridas entre 1992 e 1997:

*SELECT cg.g_city as C_City, sg.g_city as S_City, d_year, sum(lo_extendedprice*lo_discount) as revenue FROM dwg.d_customer c, dwg.f_lineorder f, dwg.d_supplier s, dwg.d_time t, dwg.d_geometry cg, dwg.d_geometry sg WHERE f.sky_customer = c.sky_customer and f.sky_supplier = s.sky_supplier and f.sky_time_orderdate = t.sky_time and c.sky_geometry = cg.sky_geometry and s.sky_geometry = cg.sky_geometry and disjoint(cg.g_map, sg.g_map)='t' And d_year >= 1992 and d_year <= 1997 group by C_City, S_City, d_year order by d_year asc, revenue desc;*

j) Qual a receita gerada por ano, agrupada pelas cidades dos compradores e dos fornecedores, onde a cidade do fornecedor não faça fronteira com a cidade do comprador, ocorridas em Dezembro de 1997:

*SELECT cg.g_city as C_City, sg.g_city as S_City, d_year, sum(lo_extendedprice*lo_discount) as revenue FROM dwg.d_customer c, dwg.f_lineorder f, dwg.d_supplier s, dwg.d_time t, dwg.d_geometry cg, dwg.d_geometry sg WHERE f.sky_customer = c.sky_customer and f.sky_supplier = s.sky_supplier and f.sky_time_orderdate = t.sky_time and c.sky_geometry = g.sky_geometry and s.sky_geometry = cg.sky_geometry And disjoint(cg.g_map, sg.g_map)='t' and d_year = 1997 and d_month = 12 group by c_city, s_city, d_year order by d_year asc, revenue desc;*

k) Qual a receita líquida gerada por ano e nação do consumidor, onde a área somada das cidades do fornecedor e do consumidor seja maior que 100.000 unidades, e o fabricante das peças seja “MFGR#1” ou “MFGR#2”:

*SELECT d_year, cg.g_nation as c_nation, sum(lo_extendedprice*lo_discount-lo_tax) as profit FROM dwg.d_customer c, dwg.f_lineorder f, dwg.d_supplier s, dwg.d_time t, dwg.d_geometry cg, dwg.d_geometry sg, dwg.d_part p WHERE f.sky_customer = c.sky_customer and f.sky_supplier = s.sky_supplier and f.sky_part = p.sky_part and f.sky_time_orderdate = t.sky_time and c.sky_geometry = cg.sky_geometry and s.sky_geometry = sg.sky_geometry and (p_mfgr = 'MFGR#1' or p_mfgr = 'MFGR#2') and area(GeomUnion(cg.g_map, sg.g_map)) >= 100000 group by d_year, c_nation order by d_year, c_nation;*

l) Qual a receita líquida gerada por ano, nação do fornecedor e categoria de peças, onde a cidade do fornecedor faça fronteira com a do consumidor, o fabricante das peças seja “MFGR#1” ou “MFGR#2” e o ano seja 1997 ou 1998:

*SELECT d_year, sg.g_nation as s_nation, p_category, sum(lo_extendedprice*lo_discount-lo_tax) as profit FROM dwg.d_customer c, dwg.f_lineorder f, dwg.d_supplier s, dwg.d_time t, dwg.d_geometry cg, dwg.d_geometry sg, dwg.d_part p WHERE f.sky_customer = c.sky_customer and f.sky_supplier = s.sky_supplier and f.sky_part = p.sky_part and f.sky_time_orderdate = t.sky_time and c.sky_geometry = cg.sky_geometry and s.sky_geometry = sg.sky_geometry and touches(sg.g_map, cg.g_map)='t' And (d_year = 1997 or d_year = 1998) and (p_mfgr = 'MFGR#1' or p_mfgr = 'MFGR#2') group by d_year, s_nation, p_category order by d_year, s_nation, p_category;*

m) Qual a receita líquida gerada por ano, cidade do fornecedor e tipo de peças, onde a cidade do fornecedor faça fronteira com a do consumidor e a soma de suas áreas seja maior que 100.000 Km2, a categoria das peças seja “MFGR#14” e o ano seja 1997 ou 1998:

*SELECT d_year, sg.g_city, p_category, sum(lo_extendedprice*lo_discount-lo_tax) as profit FROM dwg.d_customer c, dwg.f_lineorder f, dwg.d_supplier s, dwg.d_time t, dwg.d_geometry cg, dwg.d_geometry sg, dwg.d_part p WHERE f.sky_customer = c.sky_customer and f.sky_supplier = s.sky_supplier and f.sky_part = p.sky_part and f.sky_time_orderdate = t.sky_time and c.sky_geometry = cg.sky_geometry and s.sky_geometry = sg.sky_geometry and touches(sg.g_map, cg.g_map)='t' And area(GeomUnion(cg.g_map, sg.g_map)) >= 100000 and (d_year = 1997 or d_year = 1998) and p_category = 'MFGR#14' group by d_year, sg.g_city, p_category order by d_year, sg.g_city, p_category;*

Anexo III - Processo de Instalação do Servidor UAS

A primeira etapa consiste na configuração do *software web server*. Por ser o mais difundido na comunidade acadêmica, o Apache2 HTTP *server* foi *web server* utilizado neste trabalho, e portanto, a instalação compreenderá parâmetros de configuração inerentes ao mesmo.

- Editar o arquivo /etc/sysconfig/apache2 e adicionar a palavra "*rewrite*" no final da linha APACHE_MODULES, e "SSL" à linha APACHE_SERVER_FLAGS.
- Configurar o Apache para inicializar no boot através da execução do comando chkconfig -add apache2.
- Copiar e descompatcar o pacote UAS.tar.gz no *DocumentRoot* do Apache2, ou seja, na pasta /srv/www/htdocs/.
- Alterar recursivamente o proprietário e grupo da pasta /srv/www/htdocs/uas para root e www respectivamente.
- Acrescentar as linhas abaixo no arquivo /etc/apache2/httpd.conf para criar um *host* virtual. Esta configuração torna possível o acesso ao aplicativo à partir de qualquer ponto da rede local:

```
<VirtualHost *>
        ServerAdmin root@localhost
        DocumentRoot /srv/www/htdocs/uas
        ServerName uas
        ServerAlias uas
        DirectoryIndex /index.php
</VirtualHost>
```

- Criar o arquivo uas.conf em /etc/apache2/conf.d e inserir as linhas abaixo:

```
Alias /uas /srv/www/htdocs/uas
<Directory /srv/www/htdocs/uas>
        Options -Indexes Includes FollowSymLinks MultiViews
        AllowOverride None
        Order allow,deny
```

Allow from all

php_admin_value open_basedir none

</Directory>

- Reiniciar o serviço do Apache, que se tiver sido instalado no diretório padrão, pode ser efetuado através da execução do comando /etc/init.d/apache2 restart.

A segunda etapa da instalação do servidor UAS consiste na configuração do protocolo NET-SNMP.

- Definir e adicionar o nome de uma comunidade SNMP na linha rocommunity do arquivo /etc/snmp/snmpd.conf.
- Reiniciar o serviço SNMP, que se tiver sido instalado no diretório padrão pode ser efetuado através da execução do comando /etc/init.d/snmpd restart.

Anexo IV - Processo de Instalação de Clientes UAS

A instalação do cliente UAS requer que todos os pré-requisitos sejam satisfeitos, ou seja, a instalação do protocolo NET-SNMP. Por esta razão a instalação do mesmo não será descrita nesta seção.

O processo é bastante simples e constituído de apenas duas etapas.

- Definir e adicionar o nome da mesma comunidade SNMP do servidor UAS na linha rocommunity do arquivo /etc/snmp/snmpd.conf.
- Reiniciar o serviço SNMP, que se tiver sido instalado no diretório padrão pode ser efetuado através da execução do comando /etc/init.d/snmpd restart.

A segunda etapa é a execução do *script* client_add.sh, descrito no anexo V, no servidor UAS.

Localizado no diretório *home*, ou seja, /srv/www/htdocs/uas, este *script* deve ser executado conforme o exemplo a seguir e compreender a passagem de quatro parâmetros:

- ./srv/www/htdocs/uas/host_add.sh **unirio /etc/mrtg cgolap 192.168.0.1**.

  Onde:

  - unirio é o hostname do cliente UAS. Este será também nome do arquivo de configuração do cliente UAS no MRTG;

  - etc/mrtg é diretório onde serão armazenados os arquivos de saída do MRTG. Este é o diretório padrão criado na instalação do MRTG;

  - cgolap é nome da comunidade SNMP do servidor UAS;

  - 192.168.0.1 é o endereco IP do cliente UAS.

O computador onde está instalado o servidor UAS também pode ser monitorado, e para isso é necessário que o mesmo também seja configurado como um cliente UAS seguindo o procedimento descrito nesta seção.

Em razão da relação de *filesystems* montados não ser padronizada em todas as distribuições Linux, apenas o *filesystems* raíz será monitorado, pois este é o único obrigatório em todas as versões deste sistema operacional. Caso seja necessário monitorar outros *filesystems* existentes é necessário adicioná-los manualmente no arquivo /etc/snmp/snmpd.conf e configurar a SNMP MIB dskPercent correspondente no arquivo de configuração do MRTG.

Anexo V - TCPDump2NS

```
#!/bin/bash
hostname=`hostname`;
data=`date +_%d-%m-%y_%H-%M-%S`;
# Converte log do tcpdump em arquivos de texto
# A linha abaixo deve estar descomentada quando tcpdump = version 4.0.0 e libpcap =
version 1.0.0
tcpdump -r $1 | grep -F "[P.]" > networkdump
# A linha abaixo deve ser comentada quando tcpdump = version 4.0.0 e libpcap =
version 1.0.0
#tcpdump -r $1 | grep ": P " > networkdump
mtu=`ifconfig eth0 | grep MTU | awk '{print $5}' | cut -d':' -f2`

# Extrai parametros do log do tcpdump e organiza em outro arquivo
> timestamp
> sender
> receiver
> bytes

while read linha
do
echo $linha | awk '{print $1}' | cut -d. -f1 >> timestamp
echo $linha | awk '{print $3}' | cut -d. -f1 >> sender
echo $linha | awk '{print $5}' | cut -d. -f1 >> receiver
echo $linha | awk '{print $13}' >> bytes
done < networkdump

paste -d " " timestamp sender receiver bytes > extracted

# Identifica os hosts envolvidos para montar lista dos hostnames
cat extracted | awk '{print $2}' > senders
cat extracted | awk '{print $3}' > receivers
```

```
cat senders receivers > send_rec
awk '{ if ( !umArrayLinhas[$0]++ ) { print $0 } }' send_rec > hosts

#Insercao das secoes Create a Simulator Object e Finish Procedure no arquivo de
simulacao
echo "#Create a simulator object" > network_model$data.tcl
echo "set ns [new Simulator]" >> network_model$data.tcl
echo " " >> network_model$data.tcl
echo "#Define NS Trace File" >> network_model$data.tcl
echo "set arq_trace [open trace_ns w]" >> network_model$data.tcl
echo "\$ns trace-all \$arq_trace" >> network_model$data.tcl
echo " " >> network_model$data.tcl
echo "#Define NAM Trace File" >> network_model$data.tcl
echo "set arq_nam [open trace_nam w]" >> network_model$data.tcl
echo "\$ns namtrace-all \$arq_nam" >> network_model$data.tcl
echo " " >> network_model$data.tcl
echo "#Define a 'finish' procedure" >> network_model$data.tcl
echo "proc finish {} {" >> network_model$data.tcl
echo "    global ns arq_trace arq_nam" >> network_model$data.tcl
echo "    \$ns flush-trace" >> network_model$data.tcl
echo "#Close Trace Files" >> network_model$data.tcl
echo "    close \$arq_trace" >> network_model$data.tcl
echo "    close \$arq_nam" >> network_model$data.tcl
echo " " >> network_model$data.tcl
echo "#Execute nam on the trace file" >> network_model$data.tcl
echo "    exec nam trace_nam &" >> network_model$data.tcl
echo "    exit 0" >> network_model$data.tcl
echo "}" >> network_model$data.tcl

#Criacao dos Hosts no arquivo TCL
> ns_hosts
echo "#Create Nodes" >> ns_hosts
```

```
while read linha
do
echo 'set '$linha '[$ns node]' >> ns_hosts
echo '$ns at 0.0 "$'$linha' label '$linha'"' >> ns_hosts
done < hosts

#Identificacao da velocidade da placa de rede (eth0 por default) e criacao dos links
(com politica DropTail por default)
band=`/sbin/ethtool eth0 | grep Speed | awk '{print $2}' | cut -d'/' -f1`
band2=`/sbin/ethtool eth0 | grep Speed | awk '{print $2}' | cut -d'M' -f1`
negotiation=`/sbin/ethtool eth0 | grep Duplex | awk '{print $2}'`
delay=$(echo "scale=10; $mtu / ($band2 * 1024 * 1024)" | bc)
cat extracted | awk '{print $2" "$3}' > conexoes
awk '{ if ( !umArrayLinhas[$0]++ ) { print $0 } }' conexoes > conexoes_final
cat conexoes_final | awk '{print "$"$1" ""$"$2}' > conexoes
> ns_links
echo "#Connect the nodes" >> ns_links

if [ "$negotiation" = 'Full' ]
        then
                while read linha
                do
                echo '$ns' 'duplex-link' $linha $band $delay's' 'DropTail' >> ns_links
                done < conexoes
        else
                while read linha
                do
                echo '$ns' 'simplex-link' $linha $band $delay's' 'DropTail' >> ns_links
                done < conexoes
fi
```

```
#Definicao do tamanho da fila de cada no
queuesize=`ifconfig eth0 | grep txqueuelen | awk '{print $2}' | cut -d':' -f2`
> ns_queue

while read linha
do
echo $linha | awk '{print $1" queue-limit " $3" "$4}' >> ns_queue_temp
echo $queuesize >> ns_queue_size_temp
done < ns_links

paste -d " " ns_queue_temp ns_queue_size_temp >> ns_queue

#Monitoracao da fila para os links
> ns_mon_queue
echo "#Monitor the queue for the links" >> ns_mon_queue
cat ns_links | awk '{print $1 " " $2 "-op" " " $3 " " $4 " " "queuePos 0.5"}' >>
ns_mon_queue

#Conversao do tempo de envio de pacote em segundos e uniao com arquivo de
informacoes extraidas do tcpdump
> ns_time

while read linha
do
hour=$(echo $linha | awk '{print $1}' | cut -d':' -f1)
minute=$(echo $linha | awk '{print $1}' | cut -d':' -f2)
second=$(echo $linha | awk '{print $1}' | cut -d':' -f3)
echo  "($hour * 60 * 60) + ($minute * 60) + $second" | bc >> ns_time
done < extracted

ns_time_first_line=`cat ns_time | awk '(NR==1)'`
> ns_time_temp
```

```
while read linha
do
echo "($linha - $ns_time_first_line)" | bc >> ns_time_temp
done < ns_time

cat extracted | awk '{print $2" "$3" "$4}' > extracted_temp
paste ns_time_temp extracted_temp > ns_actions
rm ns_time_temp extracted_temp ns_time

#Criacao das conexoes entre Fontes e Destinos
> ns_fonts_dests
nl ns_actions > ns_actions_temp

while read linha
do
echo $linha | awk '{print "set font_tcp_"$3"_"$4"_"$1" [new Agent/TCP]"}' >>
ns_fonts_dests
echo $linha | awk '{print "set dest_tcp_"$3"_"$4"_"$1" [new Agent/TCPSink]"}' >>
ns_fonts_dests
echo $linha | awk '{print "set ftp_"$3"_"$4"_"$1" [new Application/FTP]"}' >>
ns_fonts_dests
echo $linha | awk '{print "$ns attach-agent $"$3" $font_tcp_"$3"_"$4"_"$1}' >>
ns_fonts_dests
echo $linha | awk '{print "$ns attach-agent $"$4" $dest_tcp_"$3"_"$4"_"$1}' >>
ns_fonts_dests
echo $linha | awk '{print "$ns connect $font_tcp_"$3"_"$4"_"$1"
$dest_tcp_"$3"_"$4"_"$1}' >> ns_fonts_dests
echo $linha | awk '{print "set ftp_"$3"_"$4"_"$1" [$font_tcp_"$3"_"$4"_"$1" attach-
source FTP]"}' >> ns_fonts_dests
echo $linha | awk '{print "$font_tcp_"$3"_"$4"_"$1" set packetSize_ "$5}' >>
ns_fonts_dests
```

```
echo $linha | awk '{print "$ftp_"$3"_"$4"_"$1" set maxpkts_ 1"}' >> ns_fonts_dests
echo $linha | awk '{print "$ns at "$2" \"$ftp_"$3"_"$4"_"$1" start\""}' >>
ns_fonts_dests
echo " " >> ns_fonts_dests
done < ns_actions_temp

rm ns_actions_temp
> ns_run
echo "#Call the finish procedure after 5 seconds of simulation time" >> ns_run
runtime=$(tail -1 ns_actions | awk '{print $1}')
echo "\$ns at "$runtime" \"finish\"" >> ns_run
echo " " >> ns_run
echo "#Run the simulation" >> ns_run
echo "\$ns run" >> ns_run

#Concatena arquivos e gera modelo de simulacao
cat ns_hosts >> network_model$data.tcl
cat ns_links >> network_model$data.tcl
cat ns_queue >> network_model$data.tcl
cat ns_mon_queue >> network_model$data.tcl
cat ns_fonts_dests >> network_model$data.tcl
cat ns_run >> network_model$data.tcl

#Gera arquivo de saida informando a Carga de Trabalho Normalizada por tempo,
pacotes e bytes dos hosts envolvidos
> carga_tempo_normalizada.txt

while read linha
do
fim=`cat ns_actions | grep $linha | awk '{print $1}' | tail -1`
inicio=`cat ns_actions | grep $linha | awk '{print $1}' | head -1`
echo "scale=1; $fim - $inicio" | bc > /dev/null
```

```
resultado=`echo "scale=1; $fim - $inicio" | bc`
echo $resultado"%"$linha >> carga_tempo_normalizada.txt
done < hosts

sort -n carga_tempo_normalizada.txt > carga_tempo_normalizada_temp.txt
mv carga_tempo_normalizada_temp.txt carga_tempo_normalizada.txt
sed -i "s/%/ /g" carga_tempo_normalizada.txt
cat carga_tempo_normalizada.txt | awk '{print $2": "$1}' >
carga_tempo_normalizada_temp.txt
carga_maxima=`tail -1 carga_tempo_normalizada_temp.txt | cut -d" " -f2`

echo "Carga de trabalho normalizada pelo TEMPO total em que cada host esteve
envolvido no processo durante a captura" > carga_tempo_normalizada.txt
echo "-----------------------------------------------------" >> carga_tempo_normalizada.txt

while read linha
do
no=`echo $linha | awk '{print $1}'`
carga=`echo $linha | awk '{print $2}'`
resultado2=`echo "scale=2; $carga * 100 / $carga_maxima" | bc`
echo $no" "$resultado2 >> carga_tempo_normalizada.txt
done < carga_tempo_normalizada_temp.txt

rm carga_tempo_normalizada_temp.txt
cat ns_actions | awk '{print $2" "$4}' > ns_actions2
> carga_bytes.txt

while read linha
do
cat ns_actions2 | grep $linha | awk '{print $2}' >> bytes_$linha
carga_bytes=`awk '{n+=$1} END {print n}' bytes_$linha`
echo $carga_bytes"%"$linha >> carga_bytes.txt
```

```
done < hosts

rm bytes_*
rm ns_actions2
sort -n carga_bytes.txt > carga_bytes_temp.txt
mv carga_bytes_temp.txt carga_bytes.txt
sed -i "s/%/ /g" carga_bytes.txt
cat carga_bytes.txt | awk '{print $2": "$1}' > carga_bytes_temp.txt
carga_maxima=`tail -1 carga_bytes_temp.txt | cut -d" " -f2`
echo "Carga de trabalho normalizada pelo total de BYTES transferidos por cada host
durante a captura" > carga_bytes_normalizada.txt
echo "-------------------------------------------------------" >> carga_bytes_normalizada.txt

while read linha
do
no=`echo $linha | awk '{print $1}'`
carga=`echo $linha | awk '{print $2}'`
resultado=`echo "scale=2; $carga * 100 / $carga_maxima" | bc`
echo $no" "$resultado >> carga_bytes_normalizada.txt
done < carga_bytes_temp.txt

echo " " >> carga_bytes_normalizada.txt
echo " " >> carga_bytes_normalizada.txt
rm carga_bytes_temp.txt
rm carga_bytes.txt
cat ns_actions | awk '{print $2" "$4}' > ns_actions2
> carga_pacotes.txt

while read linha
do
cat ns_actions2 | grep $linha | awk '{print $2}' >> pacotes_$linha
carga_pacotes=`wc -l pacotes_$linha`
```

```
echo $carga_pacotes"%"$linha >> carga_pacotes.txt
done < hosts

rm pacotes_*
rm ns_actions2
sort -n carga_pacotes.txt > carga_pacotes_temp.txt
mv carga_pacotes_temp.txt carga_pacotes.txt
sed -i "s/%/ /g" carga_pacotes.txt
cat carga_pacotes.txt | awk '{print $2": "$1}' > carga_pacotes_temp.txt
carga_maxima=`tail -1 carga_pacotes_temp.txt | cut -d" " -f2`

echo "Carga de trabalho normalizada pelo total de PACOTES transferidos por cada
host durante a captura" > carga_pacotes_normalizada.txt
echo "-----------------------------------------------------" >> carga_pacotes_normalizada.txt

while read linha
do
no=`echo $linha | awk '{print $1}'`
carga=`echo $linha | awk '{print $2}'`
resultado=`echo "scale=2; $carga * 100 / $carga_maxima" | bc`
echo $no" "$resultado >> carga_pacotes_normalizada.txt
done < carga_pacotes_temp.txt

cut -d"_" -f2 carga_pacotes_normalizada.txt > carga_pacotes_temp.txt
mv carga_pacotes_temp.txt carga_pacotes_normalizada.txt
echo " " >> carga_pacotes_normalizada.txt
echo " " >> carga_pacotes_normalizada.txt
rm carga_pacotes.txt
cat carga_bytes_normalizada.txt carga_pacotes_normalizada.txt
carga_tempo_normalizada.txt > carga_normalizada.txt

#Remove arquivos temporarios
```

```
rm networkdump extracted ns_queue ns_mon_queue ns_links ns_hosts ns_run
ns_fonts_dests senders receivers send_rec conexoes conexoes_final ns_queue_temp
ns_queue_size_temp timestamp sender receiver bytes hosts ns_actions
carga_tempo_normalizada.txt carga_bytes_normalizada.txt
carga_pacotes_normalizada.txt

#Move os arquivos de saida para /tmp
mv netuse_* /tmp
mv network_model_* /tmp
mv carga_normalizada.txt /tmp
```

Anexo VI - Modelo SVP

```
# Nó Mestre
Object_Desc no_mestre (
Declaration {
State Var
  Integer: Pendencias; /* 0 - sem sub-consulta pendente
                          n - n sub-consultas pendentes */
Const
  Object: NO1, NO2, NO3, NO4;
  Integer: NUM_NOS, Q1[4];
  Float: TAXA;
  Port: PORTA_IN, PORTA_OUT;
}

Initialization {
 Pendencias = 0
 NO1 = no1
 NO2 = no2
 NO3 = no3
 NO4 = no4
 NUM_NOS = 4
 Q1 = [32,32,32,32]
 TAXA = 0,89
 PORTA_IN = resposta
 PORTA_OUT = subconsulta
}

Events {
 event=Gera_Consulta(EXP, TAXA)
 condition=(Pendencias == 0)
 action= {
```

```
        int num, i, soma;
        soma = 0;
        for (i = 0; i < NUM_NOS; i++)
          {
            num = Q1[i];
            soma = soma + num;
            if (i == 0)
                msg(PORTA_OUT, NO1, num);
            else if (i ==1)
                msg(PORTA_OUT, NO2, num);
            else if (i ==2)
                msg(PORTA_OUT, NO3, num);
            else
                msg(PORTA_OUT, NO4, num);
          }
        num = soma;
        set_st("Pendencias",num);
       };
}

Messages {
 msg_rec=PORTA_IN
 action= {
        int total_pend;
        total_pend = Pendencias;
        if(total_pend > 0)
          total_pend = total_pend - 1;
        set_st("Pendencias", total_pend);
       };
}

Rewards {
```

```
}
)
Global_Rewards {
}
Independent_Chains {
}


# Nó Escravo
Object_Desc no1 (
Declaration {
State Var
  Integer: Buffer;  /* Numero buffers ocupados */
Const
  Object: MESTRE;
  Float: TAXA;
  Port: PORTA_IN, PORTA_OUT;
}

Initialization {
 Buffer = 0
 MESTRE = no_mestre
 TAXA = 1
 PORTA_IN = subconsulta
 PORTA_OUT = resposta
}

Events {
 event=Responde_Consulta(EXP, TAXA)
 condition=(Buffer > 0)
 action= {
        int num;
```

```
        num = Buffer - 1;
        msg(PORTA_OUT, MESTRE, 0);
        set_st("Buffer",num);
       };
}

Messages {
 msg_rec=PORTA_IN
 action= {
          int num;
          num = msg_data;
          num = num + Buffer;
          set_st("Buffer", num);
         };
}

Rewards {
rate_reward = ocupado
condition = (Buffer > 0)
value = 1;
}
)
```

Anexo VII - Modelo AVP

```
#Nó Mestre
Object_Desc no_mestre (
Declaration {
State Var
  Integer: Pendencias; /* 0 - sem sub-consulta pendente
                          n - n sub-consultas pendentes */
Const
  Object: NO1, NO2, NO3, NO4;
  Integer: NUM_NOS, Q1[4];
  Float: TAXA;
  Port: PORTA_IN, PORTA_OUT;
}

Initialization {
 Pendencias = 0
 NO1 = no1
 NO2 = no2
 NO3 = no3
 NO4 = no4
 NUM_NOS = 4
 Q1 = [16,16,16,16]
 TAXA = 1,12
 PORTA_IN = resposta
 PORTA_OUT = subconsulta
}

Events {
 event=Gera_Consulta(EXP, TAXA)
 condition=(Pendencias == 0)
 action= {
```

```
        int num, i, soma;
        soma = 0;
        for (i = 0; i < NUM_NOS; i++)
          {
            num = Q1[i];
            soma = soma + num;
            if (i == 0)
                msg(PORTA_OUT, NO1, num);
            else if (i ==1)
                msg(PORTA_OUT, NO2, num);
            else if (i ==2)
                msg(PORTA_OUT, NO3, num);
            else
                msg(PORTA_OUT, NO4, num);
          }
        num = soma;
        set_st("Pendencias",num);
       };
}

Messages {
 msg_rec=PORTA_IN
 action= {
        int total_pend;
        total_pend = Pendencias;
        if(total_pend > 0)
          total_pend = total_pend - 1;
        set_st("Pendencias", total_pend);
       };
}

Rewards {
```

```
}
)
Global_Rewards {
}
Independent_Chains {
}


#Nó Escravo
Object_Desc no1 (
Declaration {
State Var
  Integer: Buffer;  /* Numero buffers ocupados */
Const
  Object: MESTRE;
  Float: TAXA;
  Port: PORTA_IN, PORTA_OUT;
}

Initialization {
 Buffer = 0
 MESTRE = no_mestre
 TAXA = 1
 PORTA_IN = subconsulta
 PORTA_OUT = resposta
}

Events {
 event=Responde_Consulta(EXP, TAXA)
 condition=(Buffer > 0)
 action= {
        int num;
```

```
        num = Buffer - 1;
        msg(PORTA_OUT, MESTRE, 0);
        set_st("Buffer",num);
       };
}

Messages {
 msg_rec=PORTA_IN
 action= {
          int num;
          num = msg_data;
          num = num + Buffer;
          set_st("Buffer", num);
         };
}

Rewards {
rate_reward = ocupado
condition = (Buffer > 0)
value = 1;
}
)
```

Anexo VIII - Modelo AVP_WR

```
# Nó Mestre
Object_Desc no_mestre (
Declaration {
State Var
  Integer: Pendencias; /* 0 - sem sub-consulta pendente
                          n - n sub-consultas pendentes */
Const
  Object: NO1, NO2, NO3;
  Integer: NUM_NOS, Q1[3];
  Float: TAXA;
  Port: PORTA_IN, PORTA_OUT;
}

Initialization {
 Pendencias = 0
 NO1 = no1
 NO2 = no2
 NO3 = no3
 NUM_NOS = 3
 Q1 = [4,2,1]
 TAXA = 1
 PORTA_IN = consulta
 PORTA_OUT = consulta
}

Events {
 event=Gera_Consulta(EXP, TAXA)
 condition=(Pendencias == 0)
 action= {
        int num, i, soma;
```

```
        soma = 0;
        for (i = 0; i < NUM_NOS; i++)
          {
            num = Q1[i];
            soma = soma + num;
            if (i == 0)
               msg(PORTA_OUT, NO1, num);
            else
              if (i == 1)
                 msg(PORTA_OUT, NO2, num);
              else
                 msg(PORTA_OUT, NO3, num);
          }
        num = soma;
        set_st("Pendencias",num);
       };
}

Messages {
 msg_rec=PORTA_IN
 action= {
        int total_pend;
        total_pend = Pendencias;
        if(total_pend > 0)
          total_pend = total_pend - 1;
        set_st("Pendencias", total_pend);
       };
}

Rewards {
}
)
```

```
Global_Rewards {
}
Independent_Chains {
}
```

#Nó Escravo

```
Object_Desc no1 (
Declaration {
State Var
  Integer: Buffer;  /* Numero buffers ocupados */
Const
  Object: MESTRE, NO2, NO3;
  Float: TAXA;
  Port: PORTA_IN, PORTA_OUT, PORTA_IN2, PORTA_OUT2;
}

Initialization {
 Buffer = 0
 MESTRE = no_mestre
 NO2 = no2
 NO3 = no3
 TAXA = 1
 PORTA_IN = consulta
 PORTA_OUT = consulta
 PORTA_IN2 = ajuda
 PORTA_OUT2 = ajuda
}

Events {
 event=Responde_Consulta(EXP, TAXA)
 condition=(Buffer > 0)
```

```
action= {
        int num;
        num = Buffer - 1;
        if (num == 0)
          msg(PORTA_OUT2, NO2, 1);
        msg(PORTA_OUT, MESTRE, 0);
        set_st("Buffer",num);
       };
}

Messages {
 msg_rec=PORTA_IN
 action= {
         int num;
         num = msg_data;
         num = num + Buffer;
         set_st("Buffer", num);
        };
 msg_rec=PORTA_IN2
 action= {
         int num, no;
         no = msg_data;
         num = Buffer / 2;
         if (no != 1)
           {
            if (num < 1)
              {
                num = Buffer;
                msg(PORTA_OUT2, NO2, no);
              }
            else
              {
```

```
            if (no == 2)
              msg(PORTA_OUT, NO2, num);
            else
              msg(PORTA_OUT, NO3, num);
            num = Buffer - num;
           }
        }
       set_st("Buffer", num);
      };
}

Rewards {
rate_reward = ocupado
condition = (Buffer > 0)
value = 1;
}
)
```