UNIVERSIDADE FEDERAL DE SANTA CATARINA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Giancarlo Susin

Análise de Desempenho de um Cluster para Execução do Modelo de Previsão do Tempo ARPS

Dissertação submetida à Universidade Federal de Santa Catarina como parte dos requisitos para a obtenção do grau de Mestre em Ciência da Computação

Orientador: Prof. Dr. Paulo José de Freitas Filho

Florianópolis, Fevereiro de 2001

Análise de Desempenho de um Cluster para Execução do Modelo de Previsão do Tempo ARPS

Giancarlo Susin

Esta Dissertação foi julgada adequada para a obtenção do título de Mestre em Ciência da Computação Área de Concentração Sistemas de Computação e aprovada em sua forma final pelo Programa de Pós-Graduação em Ciência da Computação.

_____

Prof. Dr. Fernando A. Ostuni Gauthier

Banca Examinadora

_____

Prof. Dr. Paulo José de Freitas Filho (orientador)

_____

Prof. Dr. João Bosco da Mota Alves

_____

Prof. Dr. Alexandre Moraes Ramos

_____

Prof. M.Sc. Reinaldo Haas

Ser veraz – isso poucos podem. E, quem o pode, ainda não o quer! Menos que todos, porém, podem-no os bons.

Oh, esses bons! O homem bom nunca diz a verdade; ser bom dessa maneira é, para o espírito, uma doença.

Cedem, esses bons, rendem-se, seu coração repete as palavras que eles ouviram, a sua alma obedece; mas quem obedece não ouve a si mesmo!

Tudo aquilo que os bons chamam de mau deve unir-se para que nasça uma verdade; ó meus irmãos, sois suficientemente maus também para esta verdade?

O temerário ousar, a longa desconfiança, a cruel negação, o fastio, o cortar na carne viva – como é raro tudo isso unir-se! De tais sementes, porém – brota a verdade.

Ao lado da má consciência cresceu, até aqui, toda a ciência! Parti, vós que buscais o conhecimento, parti as velhas tábuas!

Friedrich Nietzsche

Aos meus pais Angelo e Tatiana,
que possibilitaram que eu
chegasse até aqui.

Sumário

SUMÁRIO ..... **I**

LISTA DE FIGURAS ..... **III**

LISTA DE TABELAS ..... **V**

RESUMO ..... **VI**

ABSTRACT ..... **VI**

1. INTRODUÇÃO E JUSTIFICATIVA **1**

1.1 DESCRIÇÃO DO PROBLEMA 2
1.2 OBJETIVOS 2
1.3 ESTRUTURA DO TRABALHO 3

2. FUNDAMENTAÇÃO TEÓRICA **4**

2.1 INTRODUÇÃO 4
2.2 CLUSTERS 4
2.2.1 Definição *4*
2.2.2 Classificação *7*
2.2.3 O Cluster Tipo "Beowulf" *9*
2.2.4 Aplicações de Clusters *10*
2.3 O MODELO DE PREVISÃO DO TEMPO ARPS 11
2.3.1 Visão Geral dos Modelos Numéricos do Tempo *12*
2.3.2 Características do ARPS *13*
2.3.3 Decomposição do Modelo ARPS *14*
2.3.4 Desbalanceamento de Carga *15*
2.3.5 Paralelização do Modelo *16*
2.3.6 Fluxograma de Execução do ARPS *17*
2.4 TECNOLOGIAS DE REDES PARA INTERCONEXÃO 18
2.4.1 O Modelo IEEE 802 ..... *18*
2.4.2 Ethernet *19*
2.4.3 Fast Ethernet *20*
2.4.4 Gigabit Ethernet *21*
2.4.5 Ethernet Comutada *22*
2.4.6 Myrinet *24*
2.4.7 SCI *27*
2.4.8 A Escolha de uma Tecnologia de Redes *28*
2.5 MODELO ANALÍTICO DE DESEMPENHO DO ARPS 28
2.5.1 Modelo do Sistema Seqüencial *29*
2.5.2 Modelo do Sistema Paralelo *30*
2.5.2.1 Decomposição Unidimensional 31
2.5.2.2 Decomposição Bidimensional 33
2.6 TRABALHOS RELACIONADOS 36

3. MATERIAIS E MÉTODOS **39**

3.1 INTRODUÇÃO 39
3.2 METODOLOGIA 39
3.3 FERRAMENTAS PARA SIMULAÇÃO 41
3.4 CONFIGURAÇÃO DO *CLUSTER* DO NURCAD 41
3.5 CONFIGURAÇÃO E EXECUÇÃO DO ARPS 42

3.6 DETERMINAÇÃO DO PARÂMETRO T_{UNIT} 43
3.7 DETERMINAÇÃO DOS PARÂMETROS T_{START} E T_{TRANS} 44

4. O MODELO DE SIMULAÇÃO 45

4.1 INTRODUÇÃO 45
4.2 TAMANHO DAS MENSAGENS 45
4.3 TEMPO DE EXECUÇÃO EM CADA NÓ 46
4.4 CARACTERIZAÇÃO DA REDE 47
4.5 COORDENADAS DOS NÓS E ENDEREÇAMENTO 49
4.6 CONTROLE DA SIMULAÇÃO 51
4.7 PERCURSO DE UMA ENTIDADE PELO SISTEMA 51
4.8 DICIONÁRIO DE DADOS DO MODELO 59

5. RESULTADOS EXPERIMENTAIS 61

5.1 INTRODUÇÃO 61
5.2 EXPERIMENTOS REALIZADOS NO *CLUSTER* DO NURCAD 61
5.2.1 Execução Seqüencial do ARPS 62
5.2.2 Execução Paralela do ARPS 62
5.3 VALIDAÇÃO DOS MODELOS DE DESEMPENHO 63
5.4 PREVISÕES DE DESEMPENHO EM CASOS REAIS 65
5.4.1 Previsões do Modelo Analítico 66
5.4.2 Previsões do Modelo de Simulação 68
5.4.3 Comparação entre as Previsões dos Modelos 71
5.4.4 Análise das Previsões do Modelo de Simulação 72

6. CONCLUSÕES 76

6.1 CONSIDERAÇÕES GERAIS 76
6.2 SUGESTÕES PARA TRABALHOS FUTUROS 77

7. REFERÊNCIAS BIBLIOGRÁFICAS 79

Lista de Figuras

Figura 1 – Arquitetura de computação em *cluster* 6

Figura 2 – Tipos de decomposição por domínio 15

Figura 3 – Fluxograma de execução do ARPS em um nó 17

Figura 4 – Relação entre os padrões IEEE 802 e RM-OSI 19

Figura 5 – Protocolo CSMA/CD 20

Figura 6 – Repetidor *Full-Duplex* 24

Figura 7 – Operação do *slack buffer* 25

Figura 8 – Formato de um quadro Myrinet 26

Figura 9 – Decomposição unidimensional com P processadores 31

Figura 10 – Decomposição bidimensional com P_x X P_y processadores 34

Figura 11 – *Cluster* do NURCAD 42

Figura 12 – Identificação dos subdomínios para uma grade 4 X 3 50

Figura 13– Fluxograma de execução do modelo de simulação 52

Figura 14 – Criação dos pacotes na etapa de inicialização 53

Figura 15 – Seção do comutador 54

Figura 16 – Lógica de encapsulamento e remontagem dos pacotes 54

Figura 17 – Controle de chegada de dados nos nós 55

Figura 18 – Seção da grade 56

Figura 19 – Criação e endereçamento de pacotes na saída de um nó 57

Figura 20 – Visão geral do modelo de simulação no Arena 58

Figura 21 – Gráfico comparando resultados medidos e obtidos dos modelos. 64

Figura 22 – Gráfico dos resultados do modelo de simulação..... 73

Lista de Tabelas

Tabela 1 – Resultados de medições da execução do ARPS no *cluster* 62

Tabela 2 – Resultados das simulações em comparação com as medições 63

Tabela 3 – Resultados das equações analíticas em comparação com as medições 63

Tabela 4 – Previsões do modelo analítico 67

Tabela 5 – Composição do T_p no modelo analítico 68

Tabela 6 – Previsões do modelo de simulação 69

Tabela 7 – Utilização das portas do comutador 70

Tabela 8 – Utilização do comutador e dos nós 70

Tabela 9 – Comparação entre as previsões dos dois modelos 71

Tabela 10 – Fatores simulados e variações explicadas por seus efeitos 74

Resumo

Este trabalho consiste em um estudo de diferentes alternativas de hardware e software disponíveis para a construção de um *cluster* de computadores de alto desempenho e baixo custo. O objetivo deste *cluster* é principalmente o processamento do modelo de previsão do tempo ARPS, que possui uma exigência de tempo máximo de processamento para que os resultados tenham utilidade prática. O desempenho desta aplicação é simulado em várias configurações, em que são variados o número de nós, a velocidade da CPU e a vazão da rede. A influência de cada fator é analisada, fornecendo importante indicação da melhor opção para a construção do *cluster*.

Abstract

This work, "Performance Analysis of a Cluster Running the ARPS Weather Forecasting Model", presents a study of different alternatives of hardware and software available to build a high performance and low cost computing cluster. The cluster's main objective is the processing of the ARPS weather forecasting model, which has a constraint in the maximum processing time to yield useable results in practice. Application's performance is simulated using configurations with differents number of nodes, CPU clock and network throughput. The influence of each factor is analysed, offering an important indication on the best choice to build the cluster.

1. Introdução e Justificativa

Os ganhos de escala acumulados ao longo de décadas de produção de chips, acentuados pela expansão do mercado de PCs, e o constante avanço tecnológico no processo de fabricação tornaram disponíveis no mercado de massa uma ampla oferta de computadores pessoais poderosos e de baixo custo. Hoje, os microprocessadores que equipam um computador pessoal estão entre os melhores que podem ser produzidos pela tecnologia atual. Além disso, software de boa qualidade, gratuito e de código-fonte aberto pode ser facilmente encontrado para a maioria das aplicações. Esta combinação de hardware e software foi aproveitada pela primeira vez para o processamento de aplicações científicas de alto desempenho em 1994, nos laboratórios do *NASA Goddard Space Flight Center*, sob a forma de um *cluster* de PCs interconectados por uma rede Ethernet, que foi apelidado de "*Beowulf*".

Este tipo de *cluster*, baseado no sistema operacional Linux e muitas vezes construído pelos próprios pesquisadores interessados, vem tendo cada vez mais aceitação na comunidade científica. Isto é devido principalmente ao seu alto desempenho e ao seu baixo custo, correspondente a uma pequena fração do preço de um supercomputador tradicional com o mesmo poder de processamento.

Várias aplicações científicas já se beneficiam deste tipo de sistema, entre as quais, na área meteorológica, destaca-se o modelo numérico de previsão do tempo A*dvanced Regional Prediction System* - ARPS. Em 3 de maio de 1999, a partir de dados de radares Doppler, o ARPS foi responsável pela primeira previsão da erupção de um tornado, em tempo real e em alta resolução, que foi realizada por um modelo numérico (CAPS, 2001a). No mesmo ano, a companhia aérea *American Airlines* iniciou a utilização de uma versão customizada do ARPS em suas atividades diárias, para previsões do tempo em escala regional afetando a área operacional de um aeroporto. Na Coréia, o ARPS foi adotado oficialmente como o sistema utilizado para previsão de tempestades (CAPS, 2001b).

Para o Estado de Santa Catarina, um modelo numérico de previsão do tempo como o ARPS pode representar um salto de qualidade na capacidade de previsão de precipitação, granizo, geadas, qualidade do ar, umidade do ar e solo, velocidade e direção do vento, etc. O ARPS tem potencial para realizar previsões com maior resolução, mais confiabilidade e maior antecedência do que as previsões disponíveis hoje.

1.1 Descrição do Problema

Para realizar previsões do tempo dentro de um prazo em que a informação tenha utilidade prática, é necessário um *cluster* de computadores com alto poder de processamento para a execução do ARPS. Suas previsões numéricas devem ser geradas e divulgadas de 5 a 10 vezes mais rapidamente que a evolução dos fenômenos meteorológicos estudados. Isto significa que os resultados do modelo numérico precisam ser obtidos em até 1 h. 30 min., aproximadamente.

Como suporte ao dimensionamento de um sistema com estes requisitos, faz-se necessário um estudo criterioso de desempenho deste tipo de *cluster*, considerando todas as variáveis envolvidas: número de nós, poder de processamento das CPUs, capacidade da rede de interconexão e carga imposta pelo ARPS, entre outras.

1.2 Objetivos

O objetivo geral deste trabalho é desenvolver modelos de desempenho de um *cluster* de computadores executando o modelo de previsão do tempo ARPS, de forma que permitam estimar qual será o tempo de execução ("*wall-clock*") de uma previsão para um determinado domínio e uma certa configuração do *cluster*.

Estes modelos devem possibilitar o estudo da influência dos diversos fatores envolvidos no desempenho do sistema, orientando o dimensionamento dos componentes do *cluster* que deverão ser adquiridos.

1.3 Estrutura do Trabalho

No Capítulo 1 – "Introdução e Justificativa", apresenta-se o tema e as justificativas para a realização deste trabalho. No Capítulo 2 – "Fundamentação Teórica", discorre-se a respeito de tecnologias e conceitos prévios necessários para o desenvolvimento deste trabalho: mostra-se um panorama dos modelos numéricos para previsão do tempo e em especial do ARPS, analisando-se seu desempenho seqüencial e paralelo; faz-se um levantamento das tecnologias de redes mais importantes para a construção de um *cluster* de alto desempenho; e apresenta-se um resumo de outros trabalhos publicados que abordam temas próximos ao deste. No Capítulo 3 – "Materiais e Métodos", relaciona-se os materiais, métodos e técnicas que fundamentam o trabalho experimental, a modelagem e as simulações realizadas. No Capítulo 4 – "O Modelo de Simulação", explica-se os detalhes da construção do modelo de simulação. No Capítulo 5 – "Resultados Experimentais", apresenta-se os dados obtidos nos experimentos e nas simulações. No Capítulo 6 – "Conclusões", traça-se as conclusões tiradas dos capítulos anteriores e sugere-se linhas para continuação do tema deste trabalho. No Capítulo 7 – "Referências Bibliográficas", são relacionadas as referências ao material bibliográfico consultado.

2. Fundamentação Teórica

2.1 Introdução

Neste capítulo são apresentados conceitos e tecnologias que fundamentam o desenvolvimento de todo este trabalho.

Inicialmente, são abordados alguns aspectos da tecnologia de *clusters* e suas aplicações. A seguir, mostra-se um panorama dos modelos numéricos para previsão do tempo e em especial do ARPS, analisando-se os fatores que afetam seu desempenho. Depois, faz-se um levantamento das tecnologias de redes mais importantes para a construção de um *cluster* de alto desempenho. Passa-se, então, à apresentação das equações do modelo analítico de desempenho do ARPS, tanto para processamento seqüencial como para processamento paralelo. Finalmente, apresenta-se um resumo de outros trabalhos publicados que abordam temas próximos ao deste.

2.2 *Clusters*

Neste sub-capítulo serão apresentadas as principais arquiteturas de software e hardware usadas para a construção de um *cluster* de alto desempenho, suas vantagens e desvantagens e algumas aplicações.

2.2.1 Definição

Um *cluster* é um sistema de processamento paralelo ou distribuído, composto de uma coleção de computadores – os nós – interconectados por uma rede de alta velocidade e agrupados de forma a trabalharem como um recurso computacional único e integrado.

Cada nó consiste em um sistema mono ou multiprocessado, com memória, dispositivos de E/S e sistema operacional. Os nós podem localizar-se em um gabinete único ou serem fisicamente separados e conectados através de uma LAN.

Os principais componentes de um *cluster* são:

- Múltiplos computadores, com uma ou mais CPUs, memória RAM e, opcionalmente, disco rígido;
- Sistemas Operacionais, de uso geral ou customizados para o *cluster*;
- Redes e Comutadores de alto desempenho;
- Protocolos e Serviços de Comunicação;
- *Middleware,* que representa os múltiplos computadores com a imagem de um sistema unificado para o usuário e garante a disponibilidade do sistema;
- Ambientes de Programação Paralela e Ferramentas: compiladores, MPI, PVM, etc.;
- Aplicações: seqüenciais, paralelas ou distribuídas.

A arquitetura típica de um *cluster* é mostrada na Figura 1.

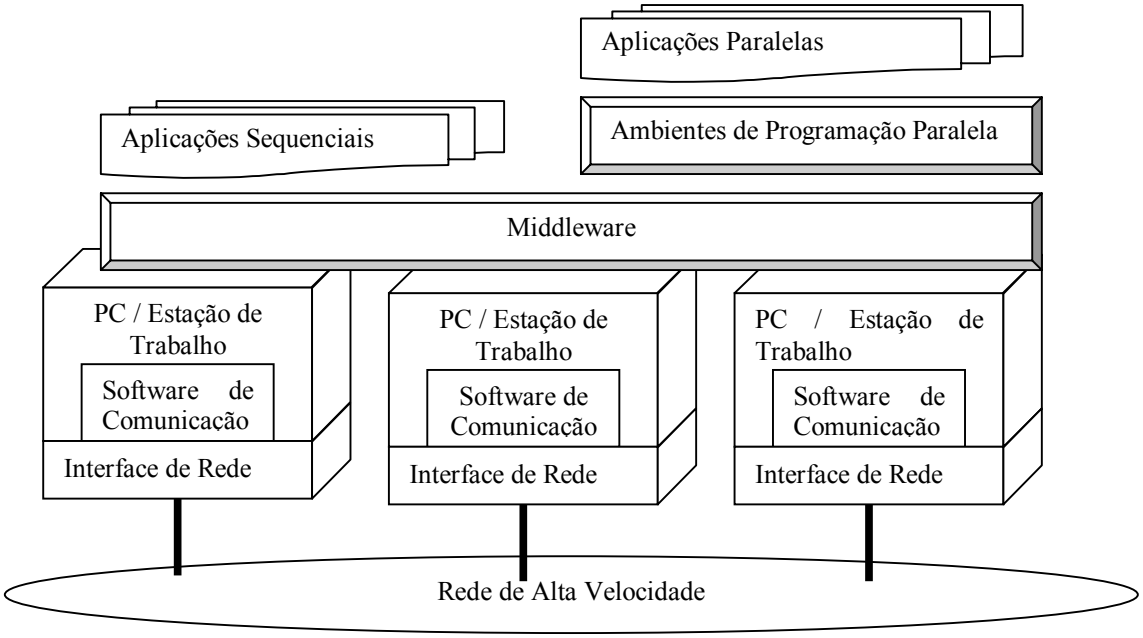


Figura 1 – Arquitetura de computação em *cluster*

O software de comunicação oferece um meio rápido e confiável de comunicação entre os nós do *cluster*. Para a comunicação entre seus nós, podem ser usados tanto os protocolos TCP/IP como outros protocolos de comunicação com baixa latência, como A*ctive Messages*, que desviam o controle do sistema operacional e podem oferecer ao usuário um acesso direto à interface de rede e remover os gargalos críticos associados à comunicação (BAKER, 1999).

Os ambientes de programação oferecem ferramentas portáveis, eficientes e fáceis de usar para desenvolvimento de aplicações. Elas incluem bibliotecas de passagem de mensagens, *debuggers* e *profilers*. Os *clusters* podem ser usados para a execução de aplicações seqüenciais ou paralelas.

Segundo BAKER (1999), algumas vantagens do processamento em *cluster* são:

- O desempenho das estações de trabalho utilizadas nos *clusters* tem aumentado dramaticamente nos últimos anos e está dobrando a cada 18 ou 24 meses. Isto provavelmente deve continuar por vários anos, com microprocessadores mais rápidos e máquinas multiprocessadas chegando ao mercado;

- À medida que novas tecnologias e protocolos são implementados em redes locais, a largura de banda entre estações de trabalho vem aumentando enquanto a latência está diminuindo;

- Os *clusters* de estações de trabalho são mais fáceis de integrar às redes existentes do que computadores paralelos especializados;

- São uma alternativa barata e prontamente disponível;

- Os *clusters* podem facilmente ser expandidos; a capacidade dos nós pode ser facilmente aumentada pelo acréscimo de memória ou microprocessadores adicionais.

Algumas desvantagens do processamento em *cluster*:

- A programação paralela não é uma técnica comumente dominada pelos pesquisadores e é bem mais complexa do que a programação seqüencial;

- Há tipos de problemas científicos que não são passíveis de processar em paralelo, portanto não são adequados para execução em *clusters*;

- A latência associada ao uso da rede de interconexão e o *overhead* da pilha de protocolos de comunicação são em geral significativamente maiores que em outras arquiteturas, como as MPP (*Massively Parallel Processing*), prejudicando o uso de *clusters* em aplicações com granularidade mais fina.

2.2.2 Classificação

Os *clusters* podem ser classificados de várias formas, conforme o critério adotado (BAKER, 1999):

a) Finalidade de Aplicação:

- Alto Desempenho – computação científica;

- Alta Disponibilidade – aplicações de missão crítica.

b) Posse dos Nós:

- *Clusters* Dedicados – o poder de processamento das estações é de uso exclusivo das aplicações submetidas ao *cluster*;

- *Clusters* Não-Dedicados – as estações são de uso pessoal e somente quando estão ociosas é que processam as aplicações do *cluster*. Isto é possível devido à constatação de que no dia-a-dia a maior parte dos ciclos de CPU de estações de trabalho não são utilizados. Através de mecanismos como migração de processos e estratégias de balanceamento de carga, as estações mantêm um desempenho interativo adequado e ainda fornecem recursos que são compartilhados com as aplicações rodando no *cluster*.

c) Hardware dos Nós:

- *Clusters* de PCs;

- *Clusters* de Estações de Trabalho;

- *Clusters* de S*ymmetric Multiprocessors* (SMPs), que são máquinas com 2 ou mais microprocessadores compartilhando todos os recursos disponíveis (barramento, memória, sistema de E/S) e rodando uma única cópia do sistema operacional.

d) Sistema Operacional dos Nós:

- Linux, Solaris, NT, AIX, Digital VMS, HP-UX, entre outros.

e) Configuração dos Nós:

- Homogêneos – todos os nós possuem arquiteturas similares e rodam o mesmo sistema operacional;

- Heterogêneos – os nós possuem diferentes arquiteturas ou rodam diferentes sistemas operacionais.

2.2.3 O *Cluster* Tipo "Beowulf"

Segundo STERLING (1995), o *cluster* tipo "Beowulf" é a denominação dada a uma classe de *clusters* construídos a partir de componentes disponíveis no mercado de massa, de alto desempenho e baixo custo. O primeiro desses *clusters* a se destacar foi o construído no *Goddard Space Flight Center* da *NASA* (*National Aeronautical and Space Administration*) em 1994, para manipulação de grandes conjuntos de dados produzidos pela pesquisa aeroespacial.

Embora os *clusters* Beowulf não obedeçam a um padrão definido, geralmente possuem algumas características em comum, conforme RIDGE (1997):

- Componentes encontrados no mercado de massa, evitando o uso de componentes fabricados sob medida;
- Os nós são dedicados;
- Utiliza uma LAN privativa;
- É de fácil replicação com componentes de fornecedores diferentes;
- Possui dispositivos de E/S escaláveis;
- Baseia-se em softwares disponíveis gratuitos e com acesso ao código fonte para customizações, como o sistema operacional Linux;
- As melhorias alcançados no design e os softwares desenvolvidos são compartilhados entre a comunidade dos usuários.

Os componentes adotados são fabricados segundo padrões largamente difundidos na indústria, beneficiando-se de preços reduzidos já que os componentes são submetidos a grande competição e produção em massa. Assim nenhum fornecedor sozinho detém os direitos sobre o produto. Vários fornecedores oferecem subsistemas idênticos, com interfaces padronizadas e aceitas universalmente.

Uma outra vantagem é que os sistemas construídos dessa forma podem evoluir no mesmo ritmo que os avanços tecnológicos, dispondo imediatamente da melhor e mais avançada tecnologia, com os melhores preços. Como conseqüência, dificilmente dois sistemas Beowulf serão idênticos quanto ao seu hardware.

A flexibilidade do sistema, montado completamente segundo as necessidades e escolhas do usuário, também é outro ponto alto. Se as necessidades mudarem com o tempo, basta selecionar novos componentes a partir de uma infinidade de opções disponíveis e reconfigurar o sistema.

2.2.4 Aplicações de *Clusters*

São inúmeras as aplicações de *clusters* de alto desempenho, entre as quais:

- Modelagem e Simulação do Clima, do Tempo e dos Oceanos;
- Química e Ciência dos Materiais;
- Eletromagnetismo e Acústica;
- Eletrônica e Nanoeletrônica;
- Dinâmica dos Fluidos;
- Mecânica Estrutural;
- Modelagem e Simulação da Qualidade Ambiental;
- Ambientes de Modelagem e Testes Integrados;
- Modelagem e Simulação de Forças Militares;
- Armas Nucleares;
- Farmacocinética;

- Genética;
- Modelagem e Simulação de Reservatórios;
- Processamento e Interpretação de Sinais Sísmicos;
- Processamento de Sinais e Imagens.

Algumas possíveis aplicações no âmbito da Universidade Federal de Santa Catarina:

- Em Meteorologia: previsões do tempo com alta resolução para o Estado de Santa Catarina utilizando o modelo ARPS;
- Em Engenharia Elétrica: análise da segurança dinâmica de sistemas elétricos de potência; planejamento da expansão e operação de sistemas hidrotérmicos; aplicações de fluxo de potência ótimo.

2.3 O Modelo de Previsão do Tempo ARPS

O *Advanced Regional Prediction System* – ARPS – é um modelo numérico tridimensional para previsão do tempo, desenvolvido no *Center for Analysis and Prediction of Storms* (CAPS) da Universidade de Oklahoma, E.U.A. Seu objetivo é gerar previsões em mesoescala (extensões até cerca de 50 km) de tempestades e seus múltiplos efeitos: inundações, ventos, granizos, tornados, turbulências para a aviação, etc. Devido à curta duração das tempestades (poucas horas) em comparação aos sistemas de tempo de larga escala (alguns dias), suas previsões numéricas devem ser geradas e divulgadas de 5 a 10 vezes mais rapidamente que a evolução dos fenômenos meteorológicos estudados. A eficácia das previsões depende fortemente do uso da computação de alto desempenho e de sistemas de comunicação velozes (SATHYE, 1997).

2.3.1 Visão Geral dos Modelos Numéricos do Tempo

O desenvolvimento de um modelo numérico hidrodinâmico complexo é uma tarefa árdua, que pode consumir dezenas de milhares de horas de trabalho ao longo de vários anos. Exige uma equipe com a participação tanto de cientistas como de engenheiros de software, além de uma metodologia bem elaborada para assimilar manutenções e aprimoramentos constantes no código do modelo.

Segundo JOHNSON (1994), um modelo do tempo é um sistema de equações diferenciais parciais formado pelas equações de Navier-Stokes mais equações para pressão, temperatura potencial, vapor de água, nuvem de chuva, água de chuva, gelo de nuvem, granizo e neve. Para possibilitar a resolução numérica do sistema de equações, o domínio físico é discretizado e as equações são transformadas para um espaço curvilíneo.

Os modelos regionais e os modelos globais do tempo estão relacionados histórica, metodológica e operacionalmente entre si (BAILLIE, 1997). Ambos resolvem as mesmas equações básicas que descrevem os fenômenos atmosféricos e são utilizados com os mesmos propósitos: previsão do tempo e do clima, assimilação e análise de dados, geração de campos meteorológicos como entrada para modelos de menor escala e pesquisa atmosférica básica.

O uso de um modelo regional ao invés de um modelo global é justificado principalmente devido ao custo computacional envolvido, que varia com o número de células do domínio e o passo de tempo. O número típico de cálculos necessários é de $O(n^4)$, onde n é uma dimensão da grade. O termo de quarta ordem reflete o refinamento necessário na dimensão temporal, pois o passo de tempo deve ser diminuído para manter a estabilidade numérica à medida que as outras dimensões são reduzidas. Como a atmosfera é rasa e o número de camadas verticais independe da resolução horizontal, o custo computacional aproxima-se melhor de uma função de n^3.

Com os modelos regionais, este enorme esforço computacional fica concentrado em uma área limitada, permitindo simulações com resolução mais alta do que seria possível usando os modelos globais. A habilidade de refinar a malha apenas em certos

subdomínios da simulação estende ainda mais a capacidade dos modelos regionais de concentrar a resolução.

Algumas diferenças entre os modelos regionais e os globais são descritas a seguir, conforme BAILLIE (1997):

- Os modelos globais buscam a solução sobre uma esfera, enquanto que os modelos regionais empregam grades cartesianas espaçadas regularmente, necessitando apenas de pequenos ajustes para compensar a distorção no arco correspondente ao domínio do modelo;

- Os modelos globais são periódicos na dimensão leste-oeste e tratam os fluxos que cruzam os pólos como casos especiais. Já os modelos regionais usam fronteiras laterais fixas ou forçadas que podem incluir alguma forma de relaxação. Muitas vezes os dados utilizados nas condições de fronteira laterais são obtidos de um modelo global;

- Os métodos numéricos dos modelos regionais podem ser mais simples que os dos modelos globais, empregando métodos de diferenças finitas sem necessidade de tratamento especial nos pólos;

- O uso destes métodos também favorece a paralelização, ao contrário do método de análise espectral usado em alguns modelos globais.

2.3.2 Características do ARPS

O modelo ARPS é projetado para ser executado eficientemente tanto em máquinas monoprocessadas como em máquinas multiprocessadas e paralelas. Seu código fonte é aberto (o*pen source*) e facilmente portável para qualquer plataforma. Sua utilização atualmente compreende as áreas de pesquisa básica, previsões operacionais e aplicações comerciais.

2.3.3 Decomposição do Modelo ARPS

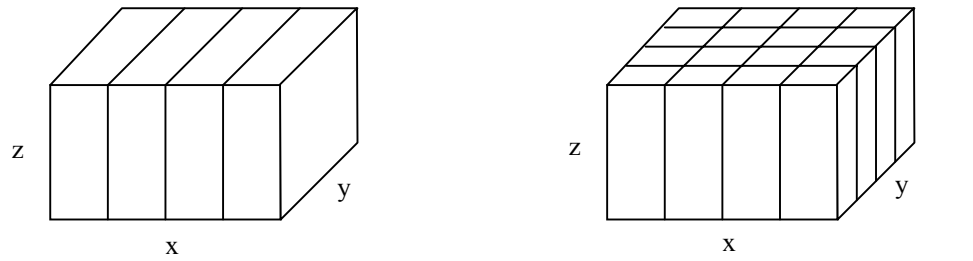
Existem duas formas de decompor o modelo para execução em múltiplos processadores: decomposição funcional e decomposição por domínio (JOHNSON, 1994).

Na **decomposição funcional**, resolve-se cada equação do modelo em um processador separado, necessitando que os resultados sejam transferidos para todos os outros processadores ao final de cada passo do cálculo.

Na **decomposição por domínio**, é atribuído um subdomínio da grade a cada processador separado e resolve-se todas as equações para ele. A comunicação entre os processadores ocorre somente para transferir os resultados referentes às fronteiras do subdomínio, ao final de cada passo do cálculo.

A melhor forma de decomposição depende da arquitetura de processamento que será adotada. Para uma arquitetura de memória distribuída, como a do *cluster*, a decomposição por domínio requer menos comunicação para a mesma quantidade de processamento.

O domínio do modelo é geralmente um paralelepípedo e seu subdomínios podem ser obtidos por seção na direção X, Y ou Z, ou em duas destas direções simultaneamente, como mostrado na Figura 2. A definição dos subdomínios para decomposição deve procurar minimizar a relação comunicação/processamento, em configurações onde a comunicação entre os nós é o fator limitante. A quantidade de dados que devem ser transferidos entre nós é proporcional à superfície do subdomínio; já a quantidade de processamento é proporcional ao volume do subdomínio. Portanto, deve-se procurar minimizar a relação superfície/volume dos subdomínios. Como os modelos de previsão do tempo geralmente tratam com grades que possuem o menor número de pontos na direção Z, a seção da grade nas direções X ou Y é mais eficiente.



Decomposição unidimensional x Decomposição bidimensional x-y

Figura 2 – Tipos de decomposição por domínio

A decomposição unidimensional gera mensagens grandes entre os subdomínios, porém em pequeno número. Já a decomposição bidimensional gera mensagens menores, porém em maior número. A escolha entre as duas vai depender da relação entre os custos de tempo de inicialização da comunicação e os custos de tempo de transmissão da tecnologia de redes utilizada.

2.3.4 Desbalanceamento de Carga

Para transferir os dados após um passo de tempo, cada processador envia os dados das bordas internas de seu subdomínio aos processadores adjacentes. Cada processador só inicia os cálculos de um novo passo de tempo após receber os dados de todos os processadores adjacentes. Esta é a única forma de sincronização existente entre os nós.

O código do ARPS permite a variação na forma da grade com o tempo. Isto possibilita a obtenção de uma melhor resolução em regiões com grandes gradientes nos campos. Assim, em alguns subdomínios haverá uma carga de processamento maior do que em outros, em algum passo de tempo.

Como o domínio não é fisicamente homogêneo, em algumas partes podem ocorrer fenômenos meteorológicos que pesarão mais para o processamento do modelo do que em outras. Por exemplo: onde houver uma nuvem será necessário processar

cálculos microfísicos de condensação, radiação e turbulência. Os processadores que não precisarem executar estes cálculos poderão ter que esperar até que um processador vizinho finalize seus cálculos e envie os resultados da sua borda adjacente. Este desbalanceamento de carga é inerente à decomposição do modelo.

2.3.5 Paralelização do Modelo

Analisando o processo de paralelização do modelo, observa-se que:

- A etapa de inicialização, quando os dados iniciais são distribuídos aos nós, não é paralelizada no modelo, somente a parte iterativa;
- Padrão de comunicação: a primeira etapa de comunicação ocorre entre um determinado processador – o *host*, que executa o código de inicialização, e os demais; na segunda etapa iniciam as integrações e somente ocorre comunicação entre os processadores vizinhos. Uma parte dos resultados é enviada de volta ao *host* em intervalos determinados;
- A quantidade de memória RAM restringe o número mínimo de processadores que pode ser usado para continuar armazenando todos os dados do problema em memória física;
- Em resoluções de meso e micro-escala, a aproximação hidrostática deixa de ter validade e as ondas acústicas precisam ser levadas em conta. A Condição de Courant-Friedrichs-Lewy (CFL) limita o tamanho máximo do passo de tempo que pode ser utilizado para que o modelo continue matematicamente estável:

$$-1 \leq \alpha . \frac{\Delta t}{\Delta x} \leq 1 \qquad (1)$$

O número α depende da velocidade da onda mais rápida do modelo. Como conseqüência desta condição, ao diminuir o espaçamento da grade (aumentando a resolução) deve-se também reduzir o tamanho do passo de tempo, aumentando o número de iterações necessárias. Por exemplo, duplicando-se a resolução no plano

X-Y faz-se com que o tempo de execução do modelo seja multiplicado por 8: duas vezes para cada uma das dimensões e mais duas vezes devido ao passo de tempo.

2.3.6 Fluxograma de Execução do ARPS

O processamento em um nó que possui outros 4 nós adjacentes obedece o fluxograma da Figura 3, para o caso de decomposição bidimensional:

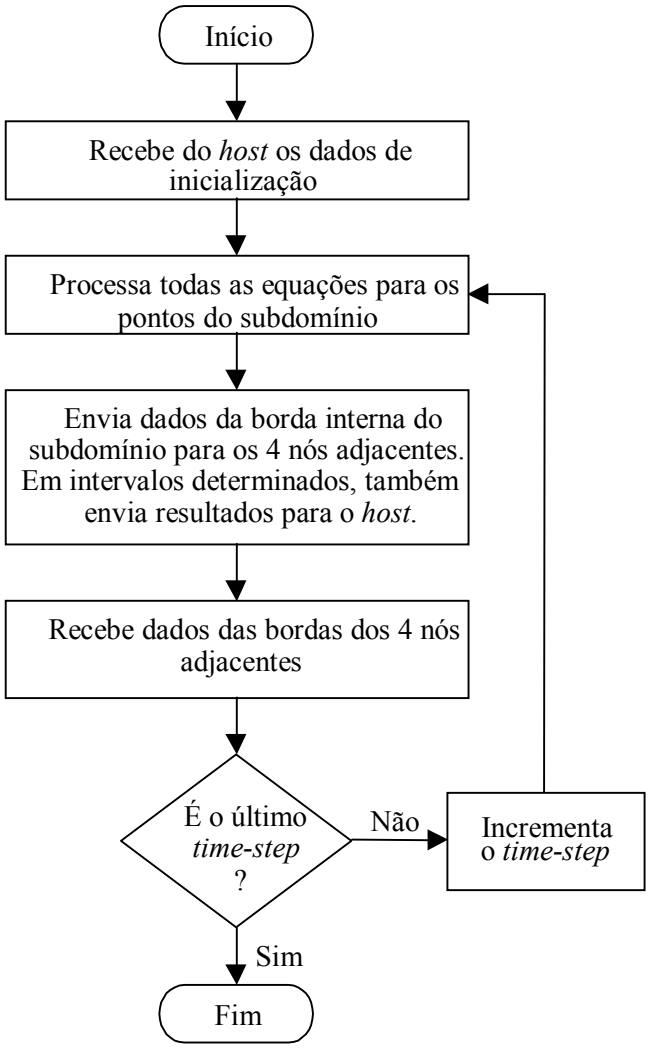


Figura 3 – Fluxograma de execução do ARPS em um nó

2.4 Tecnologias de Redes para Interconexão

Neste sub-capítulo serão descritas as tecnologias de redes mais freqüentemente usadas para interconexão dos nós de um *cluster* de alto desempenho e que estão disponíveis no mercado atual. Serão apresentadas as especificações para as camadas física e de enlace de dados que devem ser levadas em conta na criação de modelos de desempenho contendo estas tecnologias de redes.

2.4.1 O Modelo IEEE 802

Para orientar nossas análises, seguiremos a modelo definido pelo Projeto IEEE 802. O conjunto de padrões IEEE 802 foi elaborado especificamente para redes locais e baseia-se em uma arquitetura de três camadas, como descrito em SOARES (1995):

- A camada *Logical Link Control* (LLC) é responsável pelas funções de multiplexação, controle de erro e de fluxo no enlace e definição de diferentes classes de serviço;
- A camada *Medium Access Control* (MAC) é responsável pelo controle de acesso à rede. Esta configuração permite a definição de várias opções de MAC, otimizadas para as diferentes topologias de redes locais, mantendo a mesma LLC como uma interface única para os usuários da rede local;
- A camada física é responsável pelas funções de codificação/decodificação de sinais, geração e remoção de preâmbulos para sincronização e transmissão/recepção de bits.

Esta arquitetura está estreitamente relacionada ao modelo de referência OSI (*Open Systems Interconnection Reference Model* – RM-OSI) da ISO, conforme mostrado na Figura 4. As camadas LLC e MAC juntas correspondem à camada de enlace do modelo RM-OSI. A camada física corresponde à camada homônima do modelo RM-OSI.

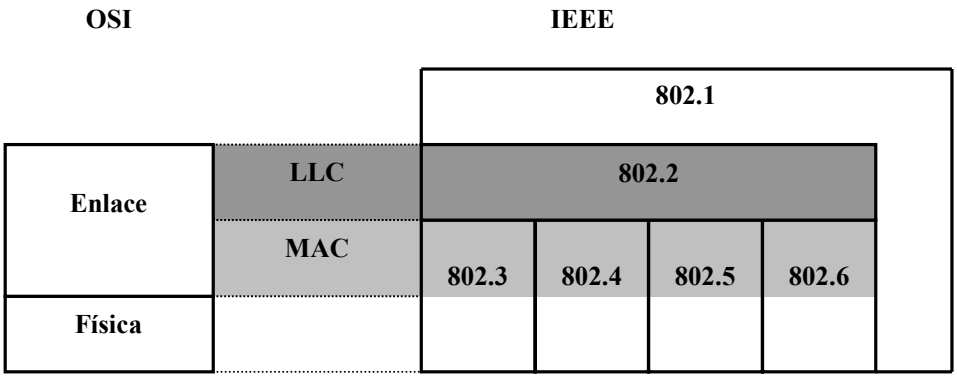


Figura 4 – Relação entre os padrões IEEE 802 e RM-OSI

O padrão IEEE 802.1 descreve o relacionamento entre os diversos padrões IEEE 802 e o relacionamento deles com o modelo de referência OSI. Também contém padrões para gerenciamento da rede e informações para ligação inter-redes. Outros padrões da série IEEE 802 serão abordados em alguns dos sub-capítulos a seguir.

2.4.2 Ethernet

Um dos primeiros padrões estabelecidos para redes locais foi o IEEE 802.3, para redes de 10 Mbps (TANENBAUM, 1997), usualmente referido como **Ethernet**, que é o nome da sua implementação mais popular.

O cabeamento mais comumente utilizado no IEEE 802.3 é o par trançado ligado a um concentrador (*hub*) central. O tamanho de seu quadro varia entre 64 bytes e 1526 bytes, com carga útil (*payload*) de até 1500 bytes por quadro. Sua duração deve ser de no mínimo 51,2 μs para permitir a detecção de colisões.

O IEEE 802.3 baseia-se no protocolo CSMA/CD (*Carrier Sense Multiple Access with Collision Detection*) para a camada MAC, esquematizado na Figura 5. Um meio compartilhado é utilizado por todas as estações, que ficam aguardando o meio estar livre para começarem a transmitir. Caso ocorra uma colisão, todas as estações tentando transmitir devem aguardar por um período de tempo aleatório, dentro de uma faixa que

cresce exponencialmente com a repetição das colisões. Este é o algoritmo de recuo binário exponencial (*binary exponencial backoff*). Com ele obtém-se um pequeno retardo quando há poucas estações em situação de colisão, mas também permite-se que as colisões sejam resolvidas em um intervalo de tempo razoável quando muitas estações colidem.

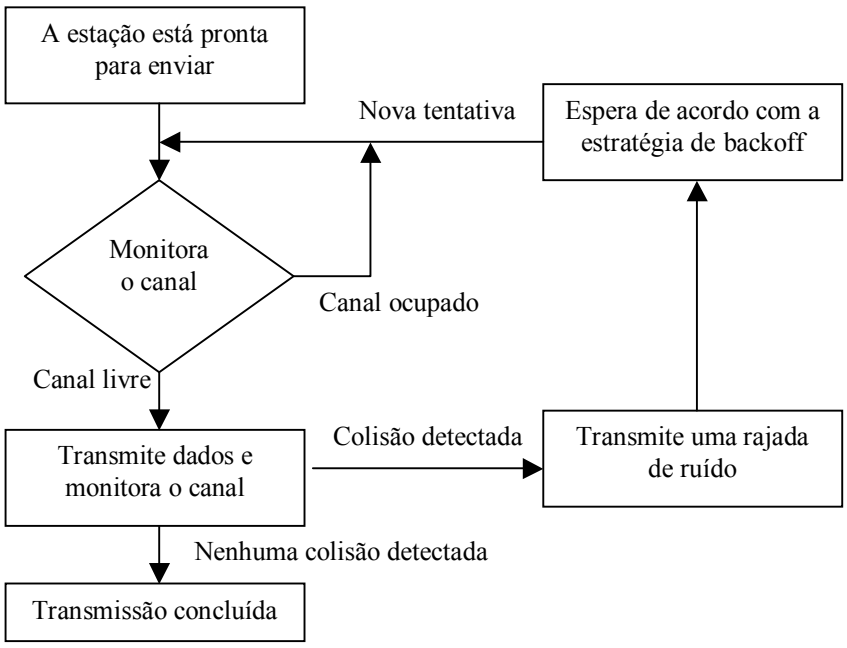


Figura 5 – Protocolo CSMA/CD

2.4.3 Fast Ethernet

Em 1995 foi aprovado o padrão IEEE 802.3u, criado para estabelecer uma rede com taxa de transmissão de 100 Mbps que mantivesse compatibilidade com o padrão IEEE 802.3 anterior, vindo a ser conhecido como **Fast Ethernet**. Utiliza-se do mesmo protocolo CSMA/CD, porém a taxas de 100 Mbps *full-duplex*, considerando-se o cabeamento de 2 pares trançados 100Base-TX.

Uma rede Fast Ethernet pode utilizar, como elemento centralizador, um concentrador (*hub*) ou um comutador. No caso do comutador, há *buffers* de entrada que armazenam os quadros a serem transmitidos e, interligando as portas, há um barramento

de alta velocidade que permite que várias estações transmitam ao mesmo tempo, reduzindo o problema das colisões.

2.4.4 Gigabit Ethernet

O padrão IEEE 802.3z foi publicado em junho de 1998 como um suplemento ao padrão ANSI/IEEE 802.3 "fifth edition". O padrão IEEE 802.3z compreende tanto a nova forma de operação *full-duplex* como a clássica CSMA/CD, para a subcamada MAC. A MAC *full-duplex* permite transmissões ponto-a-ponto bidirecionais simultâneas, com cada estação transmitindo à taxa de dados nominal.

O protocolo CSMA/CD clássico foi mantido para a operação a 1000 Mbps usando uma técnica chamada *carrier extension*. Com isto, foi superada a limitação do algoritmo de detecção de colisão, que exige que o retardo de propagação de ida e volta (*round trip delay*) entre duas estações quaisquer não exceda o tempo necessário para transmitir um quadro com o menor tamanho permitido. O tempo destinado para o tempo de portadora mínimo e o intervalo de transmissão mínimo (*slot time*) é aumentado de 64 bytes para 512 bytes, sem alterar o tamanho mínimo do quadro de 64 bytes. Quadros menores que 512 bytes são aumentados com um novo campo chamado extensão de portadora (*carrier extension)*, posicionado em seguida ao campo CRC; quadros maiores que 512 bytes não são estendidos (GEA, 1998).

Para melhorar o desempenho do protocolo CSMA/CD, foi definida uma característica opcional chamada *frame bursting*, que permite a uma estação concatenar quadros adicionais na mesma rajada da portadora, até que um temporizador de rajada se esgote. Com isso obtém-se uma significativa melhora na utilização da largura de banda disponível.

Para a operação *full-duplex*, a subcamada MAC baseia-se no fato de que o enlace de comunicação serial suporta transmissões simultâneas bidirecionais sem interferência entre os sinais enviados e os recebidos. Essencialmente, o protocolo CSMA/CD é desabilitado e, já que não existe restrição de retardo de propagação de ida

e volta, a técnica de *carrier extension* não é mais necessária. Para poder transpor um enlace qualquer, a única limitação passa a ser a camada física (FRAZIER, 1998).

A especificação da operação *full-duplex* também traz um mecanismo para controle de fluxo. Se ocorrer um congestionamento em um enlace, o receptor pode inibir o envio de novos quadros emitindo um quadro PAUSE, instruindo assim o transmissor a deter as próximas transmissões durante um certo período de tempo.

2.4.5 Ethernet Comutada

Um comutador (ou *switch*) é um equipamento concentrador capaz de efetuar várias conexões simultâneas entre suas portas de entrada e saída, segundo o endereço que consta no cabeçalho de cada quadro que trafega pela rede. Tipicamente, ele tem uma latência um pouco maior que um repetidor, o que pode representar uma limitação em aplicações paralelas fortemente acopladas. Mas, como a latência de aplicações que usam o *kernel* do sistema operacional para enviar mensagens é da ordem de 100 μs, em sistemas práticos a latência relacionada à aplicação predomina sobre a latência relacionada ao comutador (HAWICK, 1999b).

Segundo SOARES (1995), os comutadores podem ser classificados como:

- **Cut-Through**: lê apenas a informação do cabeçalho MAC antes de começar o processamento; é muito rápido, porém quadros ruins são repassados adiante;
- **Store-and-Forward**: lê todo o quadro na memória e verifica se há erros; só depois o quadro é transmitido. É mais lento que o Cut-Through mas não repassa quadros ruins;
- **Error-Free Cut-Through**: lê tanto o endereço quanto a seqüência de verificação de cada quadro. O quadro é repassado imediatamente, como no Cut-Through. Porém, se um quadro ruim foi enviado, o comutador reconfigura a porta para comutação Store-and-Forward. Depois que os erros baixam de um certo limiar, a porta volta a operar como Cut-Through.

A Ethernet Comutada (*Switched Ethernet*) é definida no padrão IEEE 802.1D, e pode ser aplicada a redes de 10, 100 ou 1000 Mbps. Os quadros são filtrados com base no endereço MAC e enviados de uma porta a outra através da malha de comutação, que pode ser um *cross-bar* ou um barramento de alta velocidade. Para que ele opere na vazão (*throughput*) máxima, a largura de banda da malha de comutação deve ser igual à soma das larguras de banda de todas as portas conectadas. No caso de múltiplas estações transmitirem quadros simultaneamente para uma mesma porta de saída, é utilizado o sistema de controle fluxo PAUSE. Assim, cada comutador deve possuir uma função de controle de acesso ao meio (MAC), um *buffer* de alta velocidade para os quadros, uma interconexão de alta velocidade, uma memória da tabela de endereçamento e função de aprendizagem de roteamento, aumentando seu custo em relação a um simples repetidor (CHRISTENSEN, 1998).

O repetidor tradicional usado em redes Ethernet não contém nenhuma função de controle na subcamada MAC e nem *buffers*; a arbitragem das transmissões é feita nas estações pelos algoritmos distribuídos CSMA/CD half-duplex e *Binary Exponential Backoff*. A partir do advento da Gigabit Ethernet, uma nova opção tornou-se disponível: o Repetidor *Full-Duplex* (*Full-Duplex Repeater* – FDR). Em um FDR são acrescentados uma função MAC full-duplex, *buffers* e arbitragem de transmissão centralizada, como mostrado na Figura 6. Entre as estações e o repetidor, o enlace é full-duplex, enquanto que dentro do repetidor um barramento de 1 Gbps é compartilhado entre as estações. Um mecanismo de arbitragem como o *round robin* (MENASCÉ, 1994) determina o próximo quadro que será transferido dos *buffers* de entrada do repetidor para todas as portas de saída.

A filtragem dos quadros ocorre somente nas estações receptoras. Se houverem múltiplas estações tentando transmitir, o mecanismo de controle de fluxo PAUSE (padrão IEEE 802.3x) é usado para limitar a taxa de envio a 1 Gbps no máximo. Ao receber um quadro PAUSE, a estação termina a transmissão que está fazendo e entra em estado de pausa durante um período determinado. Se durante esse período a estação receber outro quadro PAUSE, esse tempo pode ser estendido, reduzido ou encerrado. Com isso, o repetidor evita o estouro em seus *buffers* de entrada e o compartilhamento do barramento de 1 Gbps ocorre sem nenhuma colisão.

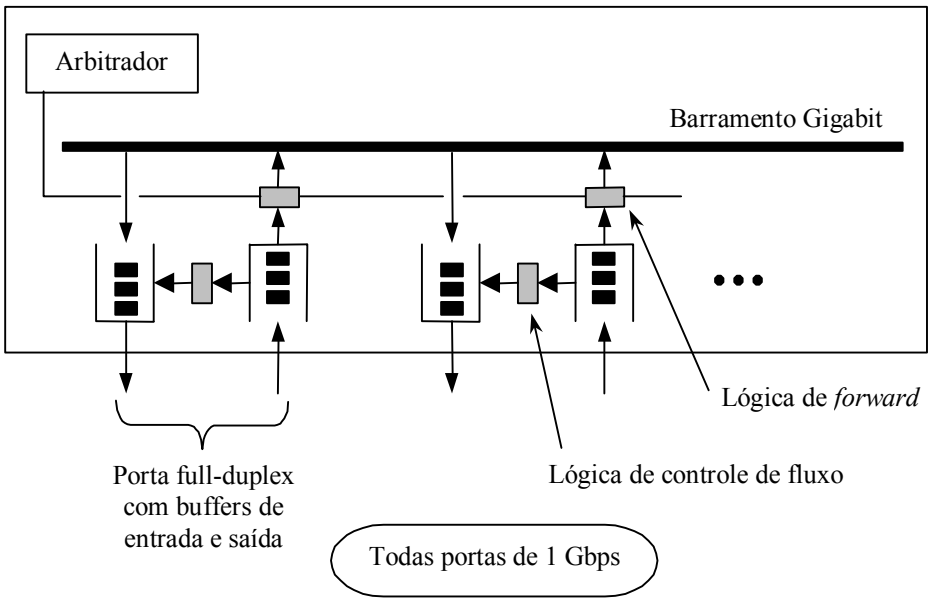


Figura 6 – Repetidor *Full-Duplex*
Destacam-se o arbitrador centralizado, os *buffers* de entrada e saída e o mecanismo de controle de fluxo.

A estas taxas de transmissão, o gargalo desloca-se para outros pontos do sistema. Mesmo se tivéssemos uma rede de interconexão ponto-a-ponto de alta velocidade entre todos os nós, a transmissão simultânea a partir de vários nós para um único nó rapidamente esgotaria a capacidade do barramento interno da CPU.

2.4.6 Myrinet

Uma rede Myrinet, especificada no padrão ANSI/VITA 26-1998, é composta de enlaces ponto-a-ponto, *full-duplex*, que conectam estações e comutadores. Os comutadores podem ser conectados a outros comutadores ou diretamente a estações usando uma topologia qualquer. O padrão para cabeamento define que para cada enlace o comprimento máximo de cabo é de 25 m, formado por 18 pares trançados, 9 em cada sentido. A taxa de dados de um canal Myrinet é de 1280 Mbps.

O controle de fluxo é realizado pelo receptor, injetando os símbolos de controle STOP e GO no canal de sentido oposto do enlace. O receptor possui um *buffer* (*slack buffer*) contendo dois limiares, conforme a Figura 7. Quando o *buffer* é preenchido até ultrapassar o limiar STOP, um símbolo de controle STOP é gerado para fazer com que o fluxo pare antes que o *buffer* transborde. À medida que o *buffer* vai sendo esvaziado e o nível atinge o limiar GO, um símbolo de controle GO é gerado para fazer com que o fluxo recomece. As posições do *buffer* entre os limiares STOP e GO geram uma espécie de histerese, assegurando que os símbolos de controle não consumirão uma largura de banda excessiva (BODEN, 1995).

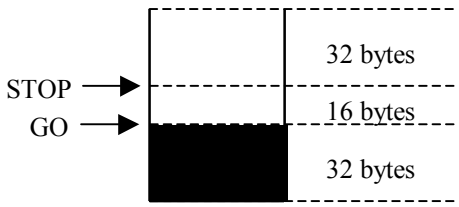


Figura 7 – Operação do *slack buffer*

Quando um quadro Myrinet entra em um comutador, o primeiro byte do cabeçalho é lido e determina a porta de saída. Logo após, este byte é descartado do quadro. Quando um quadro entra em uma interface de rede de uma estação, o primeiro byte identifica o tipo de quadro, que pode ser: um quadro de gerenciamento, um quadro de mapeamento, um quadro contendo um datagrama IP, etc. O bit mais significativo de cada byte do cabeçalho distingue entre bytes destinados para estações e bytes destinados para comutadores. Esta redundância permite às interfaces detectarem e resolverem falhas de roteamento, além de simplificar o processo de mapeamento da rede.

O formato de um quadro Myrinet é mostrado na Figura 8. O campo de dados de um quadro Myrinet possui tamanho arbitrário, portanto não há um MTU (*Maximum Transfer Unit*) definido. Assim, ele pode carregar qualquer tipo de pacote sem a necessidade de uma camada de adaptação.

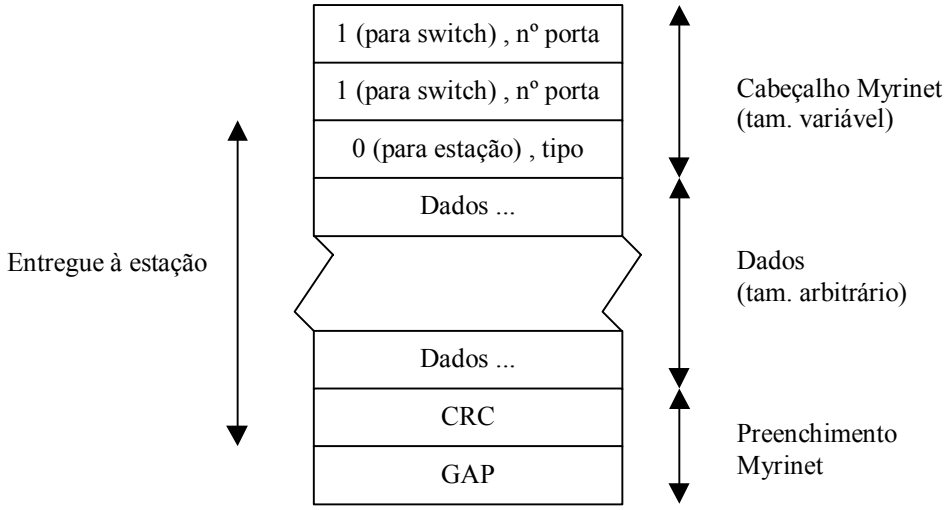


Figura 8 – Formato de um quadro Myrinet

Os comutadores Myrinet utilizam o roteamento *block-cut-through* (BODEN, 1995), mais rápido, pois podem contar com o controle de fluxo sendo feito pelas estações. A topologia dos comutadores é em malha bidimensional; com isso, sua capacidade agregada cresce com o número de nós, já que vários quadros podem transitar concorrentemente ao longo de diferentes caminhos.

O Myrinet Control Program (MCP) é carregado pelo *driver* de dispositivo quando a estação é inicializada e interage concorrentemente com a própria estação e com a rede. O MCP realiza continuamente: operações de mapeamento e monitoração (remapeamento), que tornam a rede auto-configurável e auto-recuperável; seleção de rotas; e *multicast store-and-forward*. Uma das interfaces das estações em cada rede é selecionada para mapear a rede, enviando quadros de mapeamento para as outras estações e para si mesma. O mapa da rede é distribuído pela interface mapeadora para todas as outras interfaces. Então cada interface calcula as rotas dela mesma para todas as outras interfaces.

A API (*Application Programming Interface*) Myrinet permite melhorar o desempenho da transferência de dados para a interface, através de uma área de memória alocada para transferência de dados, o que evita as chamadas às funções de comunicação do sistema operacional que fazem múltiplas cópias dos dados.

2.4.7 SCI

A tecnologia SCI (*Scalable Coherent Interface*), definida pelo padrão IEEE 1596-1992, fornece suporte de hardware a memória compartilhada distribuída, permitindo a cada nó do *cluster* definir regiões de memória compartilhada. Todas as regiões definidas pertencem ao espaço de endereçamento global de 64 bits da SCI; desta forma, a comunicação é realizada transparentemente escrevendo e lendo da memória. A comunicação real fica a cargo da interface de rede SCI sempre que uma região compartilhada remota é acessada pelo usuário. Problemas típicos da memória compartilhada distribuída, como falso compartilhamento, são evitados porque as regiões compartilhadas não são replicadas, mas verdadeiramente compartilhadas apesar da memória estar distribuída fisicamente (OLIVEIRA, 2000).

Conforme ÁVILA (1999), pode-se destacar as seguintes características da tecnologia SCI:

- Utiliza como modelo de programação a memória compartilhada, ao invés da passagem de mensagens: um processo pode ter mapeado em seu espaço de endereçamento um segmento de memória fisicamente localizado em outro nó;
- Sua configuração típica é em anel, com dois canais em cada placa, um de entrada e um de saída;
- Seu desempenho nominal prevê uma vazão de 4 Gbps (500 MBytes/s) e uma latência de 3 μs;
- Não necessita comutador, pois as próprias placas já fazem o roteamento e o acesso às memórias remotas.

2.4.8 A Escolha de uma Tecnologia de Redes

HAWICK (1999a) faz uma comparação entre algumas tecnologias de redes utilizadas na construção de *clusters* de alto desempenho, quanto a desempenho e custo por nó instalado. Ele conclui que, para aplicações que não necessitam da largura de banda na faixa dos Gbps, a Fast Ethernet é a escolha mais lógica. Mas se a largura de banda da ordem de Gbps é necessária, na decisão deve-se pesar: a largura de banda e a latência atingíveis; o custo aproximado por nó; e se a tecnologia está disponível no mercado de massa e é oferecida por vários fabricantes.

A escolha de uma tecnologia de redes não deve ser feita de forma isolada, e sim baseada em uma análise abrangente das necessidades da aplicação, em que o desempenho é apenas uma das variáveis envolvida.

2.5 Modelo Analítico de Desempenho do ARPS

JOHNSON (1994) estabeleceu um modelo analítico que permite prever o desempenho da execução paralela do ARPS em *clusters* de computadores. O modelo analítico desenvolvido pressupõe as seguintes simplificações, válidas tanto no caso seqüencial como no caso paralelo:

- Não há alteração da resolução da grade em função do tempo;
- Nenhum cálculo microfísico é realizado nos subdomínios e a carga de processamento é igual em todos eles;
- Todos os tipos de operações de ponto flutuante levam o mesmo tempo para serem executados;
- O tempo para calcular uma etapa do modelo é diretamente proporcional ao número de operações de ponto flutuante executadas.

A seguir, serão apresentadas as equações do modelo analítico, tanto para processamento seqüencial como para processamento paralelo.

2.5.1 Modelo do Sistema Seqüencial

Uma sessão de execução do modelo ARPS constitui-se de duas etapas:

a) Etapa de inicialização, com duração T_{init} , quando cada processador recebe os dados com as condições iniciais e preenche os *arrays*;

b) Etapa de iteração, com duração T_{iter} , quando são conduzidos os cálculos iterativos que vão avançando a cada passo de tempo.

O tempo total de execução seqüencial pode ser expresso como:

$$T_{seq} = T_{init} + T_{iter} \tag{2}$$

Durante a fase de inicialização, são executadas 71 somas, 97 multiplicações, 5 divisões e 1 exponenciação por ponto da grade, totalizando 174 operações de ponto flutuante.

Durante a fase iterativa, a cada passo de tempo são executadas 725 somas, 805 multiplicações e 44 divisões por ponto da grade, totalizando 1574 operações de ponto flutuante.

Sendo N_x , N_y e N_z o número de pontos nas direções X, Y e Z, respectivamente, e N_t o número de passos de tempo, então:

$$T_{seq} = \left(174.N_x.N_y.N_z + 1574.N_x.N_y.N_z.N_t\right).t_{unit} \tag{3}$$

onde t_{unit} é o tempo típico necessário para executar uma operação de ponto flutuante genérica, tal como uma adição ou uma multiplicação.

2.5.2 Modelo do Sistema Paralelo

Ao desenvolver o modelo para o sistema multiprocessado, deve-se levar em conta que:

- Há código não paralelizável no modelo;
- Há *overhead* de comunicação entre os processadores adjacentes;
- Há *overhead* de software devido aos cálculos redundantes nas bordas externas dos subdomínios.

No modelo multiprocessado assume-se mais algumas simplificações, além das relacionadas no sub-capítulo 2.5:

- Todas as máquinas são dedicadas e com a mesma velocidade de processamento;
- Todos os subdomínios possuem o mesmo número de bordas, cujos valores calculados devem ser transferidos para os processadores adjacentes;
- O tempo de inicialização de uma comunicação entre processadores é independente do tamanho da mensagem transmitida;
- Não ocorre nenhum congestionamento ou redução na vazão da rede quando os processadores enviam suas mensagens ao mesmo tempo, ou seja, o tempo de comunicação é independente do número de processadores tentando transmitir;
- O tempo para inicialização do modelo é muito pequeno em relação ao tempo para as iterações entre os processadores e pode ser desprezado;
- Não ocorre superposição entre os períodos de computação nos processadores e os períodos de transmissão de dados pela rede – a cada instante o sistema ou está calculando ou está comunicando.

2.5.2.1 Decomposição Unidimensional

Consideremos primeiro o caso da decomposição unidimensional na direção X, como na Figura 9.

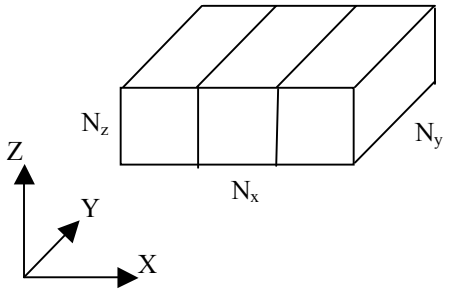


Figura 9 – Decomposição unidimensional com P processadores

O tempo de execução do código paralelizado T_p compõe-se de uma parcela devida ao tempo de computação T_{calc} e uma parcela devida ao tempo de comunicação T_{comm}. Para o caso de uma grade com P subdomínios de tamanho $N_x.N_y.N_z / P$:

$$T_P = T_{calc} + T_{comm} \tag{4}$$

O tempo de computação T_{calc} possui uma parcela referente à inicialização T_{init}, uma parcela referente às iterações T_{iter} e uma parcela referente aos cálculos redundantes decorrentes da paralelização T_{dup}:

$$T_{calc} = T_{init} + \frac{T_{iter}}{P} + T_{dup} \tag{5}$$

O número de operações de ponto flutuante executadas durante a fase de inicialização é o mesmo que o do caso seqüencial.

Durante a fase iterativa, a cada passo de tempo são executadas em cada processador 1574/P operações de ponto flutuante por ponto da grade.

O número de operações de ponto flutuante redundantes por borda adjacente é de 669 operações (295 somas e 374 multiplicações). Como no caso unidimensional cada

subdomínio possui duas bordas adjacentes, com exceção dos subdomínios das extremidades, uma estimativa superior para o número de operações redundantes é:

$$N_{dup} = 1338.N_y.N_z.N_t \quad (6)$$

Expressando em função de t_{unit} , que é o tempo típico para executar uma operação de ponto flutuante genérica:

$$T_{calc} = \left(174.N_x.N_y.N_z + \frac{1574.N_x.N_y.N_z.N_t}{P} + 1338.N_y.N_z.N_t \right).t_{unit} \quad (7)$$

O tempo de comunicação T_{comm} é composto de uma parcela correspondente ao tempo de inicialização T_{base} e uma parcela correspondente ao tempo de comunicação iterativa T_{itcom} :

$$T_{comm} = T_{base} + T_{itcom} \quad (8)$$

Com base em dados empíricos de transmissão de uma rede Ethernet utilizando um concentrador (*hub*) como elemento centralizador, em que o número de processadores tentando transmitir foi variado de 2 a 16, obteve-se:

$$T_{base} = (P-2).\left[35.t_{start} + 41.N_y.N_z.\left(\frac{N_x}{P} + 2 \right).t_{trans} \right] + 2.\left[35.t_{start} + 41.N_y.N_z.\left(\frac{N_x}{P} + 1 \right).t_{trans} \right] \quad (9)$$

onde t_{start} é o tempo para inicializar uma transmissão e t_{trans} é a taxa de transmissão da rede, em segundos por palavra.

O número de campos de prognóstico transferidos de um processador a outro por ponto da grade é 11. Para um subdomínio com duas bordas adjacentes, sem levar em conta o efeito do número de processadores tentando transmitir (gargalos ou redução de vazão):

$$T_{itcom} = 2.(t_{start} + 11.N_y.N_z.t_{trans}).N_t \quad (10)$$

Assim,

$$T_{comm} = (P-2).\left[35.t_{start} + 41.N_y.N_z.\left(\frac{N_x}{P}+2\right).t_{trans}\right] +$$
$$+2.\left[35.t_{start} + 41.N_y.N_z.\left(\frac{N_x}{P}+1\right).t_{trans}\right] + 2.N_t.t_{start} + 22.N_y.N_z.N_t.t_{trans} \quad (11)$$

Levando-se em conta que o tempo de inicialização do ARPS é geralmente pequeno em comparação à parte iterativa do código, T_{init} e T_{base} podem ser desconsiderados. Assim,

$$T_P = \left(\frac{1574.N_x.N_y.N_z.N_t}{P} + 1338.N_y.N_z.N_t\right).t_{unit} +$$
$$+2.N_t.t_{start} + 22.N_y.N_z.N_t.t_{trans} \quad (12)$$

Por analogia, para a decomposição unidimensional na direção Y:

$$T_P = \left(\frac{1574.N_x.N_y.N_z.N_t}{P} + 1338.N_x.N_z.N_t\right).t_{unit} +$$
$$+2.N_t.t_{start} + 22.N_x.N_z.N_t.t_{trans} \quad (13)$$

2.5.2.2 Decomposição Bidimensional

A análise para o caso de decomposição no plano X-Y não foi apresentada em JOHNSON (1994), por isso passamos a deduzi-la como segue.

Consideremos uma grade com subdomínios em forma de prismas retos, com base retangular de lados a e b, e altura N_z , como na Figura 10.

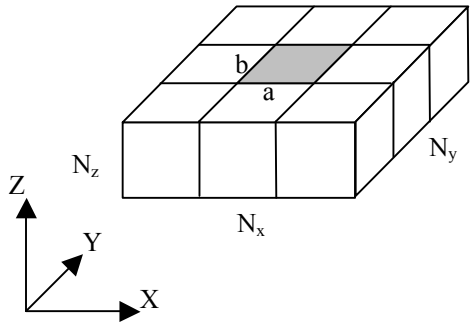


Figura 10 – Decomposição bidimensional com P_x X P_y processadores

O número total de subdomínios P é dado por:

$$P = P_x . P_y \tag{14}$$

onde P_x é o número de processadores na direção X e P_y é o número de processadores na direção Y.

Os lados da base do prisma são dados por:

$$a = \frac{N_x}{P_x} \tag{15}$$

e

$$b = \frac{N_y}{P_y} \tag{16}$$

O número de operações redundantes nas bordas externas é:

$$N_{dup} = (2.a.N_z + 2.b.N_z).N_t .Op$$

$$= 2.\left(\frac{N_x}{P_x} + \frac{N_y}{P_y}\right).N_z.N_t.Op$$

$$= \frac{2.(N_x.P_y + N_y.P_x).N_z.N_t.Op}{P} \tag{17}$$

onde Op é o número de operações de ponto flutuante para o cálculo de um operador Jacobiano, igual a 669 operações.

Para $P_x > 1$ e $P_y > 1$:

$$N_{dup} = \frac{1338.(N_x.P_y + N_y.P_x).N_z.N_t}{P} \quad (18)$$

Para o caso particular $P_y = 1$ (decomposição unidimensional) não há operações nas bordas na direção Y:

$$N_{dup} = 1338.N_y.N_z.N_t \quad (19)$$

E analogamente para $P_x = 1$:

$$N_{dup} = 1338.N_x.N_z.N_t \quad (20)$$

Assim, o tempo de computação T_{calc} para o caso bidimensional resulta em:

$$T_{calc} = \left[174.N_x N_y N_z + \frac{1574.N_x.N_y.N_z.N_t + 1338.(N_x.P_y + N_y.P_x).N_z.N_t}{P} \right].t_{unit} \quad (21)$$

Considerando-se que cada subdomínio possui outros quatro subdomínios adjacentes (dois na direção X e mais dois na direção Y), o tempo de comunicação iterativa é:

$$T_{itcom} = \left[4.t_{start} + 22.(N_y + N_x).N_z.t_{trans} \right] N_t \quad (22)$$

Desconsiderando-se o termo T_{base} na equação (8) da mesma forma que foi feito no sub-capítulo 2.5.2.1, tem-se que $T_{comm} = T_{itcom}$. Assim, a equação (4) reduz-se a:

$$T_P = T_{calc} + T_{itcom} \quad (23)$$

Finalmente, o tempo de execução do código paralelizado T_p para o caso bidimensional pode ser expresso como:

$$T_P = \left[174.N_x N_y N_z + \frac{1574.N_x.N_y.N_z.N_t + 1338.(N_x.P_y + N_y.P_x).N_z.N_t}{P} \right].t_{unit} + \\ + 4.N_t.t_{start} + 22.(N_y + N_x).N_z.N_t.t_{trans} \quad (24)$$

Para o caso particular em que $P_y = 1$ ou $P_x = 1$ (decomposição unidimensional), aplica-se as equações (12) ou (13).

2.6 Trabalhos Relacionados

Aqui serão descritos brevemente alguns trabalhos de outros autores que seguiram linhas de pesquisa próximas da adotada neste trabalho.

JOHNSON (1994) descreve o modelo do tempo ARPS, apresenta as grandezas físicas envolvidas e as equações que descrevem os fenômenos meteorológicos. Discute técnicas para otimizar a paralelização deste modelo. Desenvolve modelos analíticos de desempenho do ARPS em máquinas seqüenciais e em multicomputadores. Compara o desempenho previsto com os resultados de experimentos em um *cluster* de estações IBM RS6000 com 16 nós, interligados por rede Ethernet. Conclui que nesta configuração o gargalo do sistema é a rede de interconexão e prevê o desempenho futuro com redes mais velozes (hoje já disponíveis).

HOE (1999) analisa o desempenho de um *cluster* denominado "Hyades" com 16 nós, dedicado para simulações climáticas do GCM (*General Circulation Model*) desenvolvido no *Massachusetts Institute of Technology*. Cada nó é construído com um SMP contendo 2 microprocessadores Pentium 400 MHz e 512 Mbytes RAM, conectado a uma malha de comutação de alto desempenho Arctic capaz de transferir 150 MBytes/s por enlace. Para reduzir o *overhead* de software e utilizar melhor a capacidade do hardware, foi desenvolvida uma biblioteca de comunicação (API) específica para esta aplicação. O desempenho do *cluster* foi analisado através de modelos analíticos, investigando os gargalos do sistema e o seu desempenho em comparação a outras plataformas. Dado o alto desempenho obtido e sua disponibilidade total ao pesquisador, o *cluster* com esta arquitetura foi denominado um "supercomputador pessoal".

BRIGHTWELL (2000) faz uma exposição do projeto "Cplant", desenvolvido nos *Sandia National Laboratories* para pesquisa de software paralelo em máquinas de memória distribuída, utilizando passagem de mensagens. Foi construído um *cluster* baseado em Linux com 400 nós, a partir de componentes disponíveis no mercado de massa. Seu desempenho foi avaliado em comparação com o supercomputador Sandia/Intel TeraFLOPS, uma máquina capaz de chegar a 3,2 TFLOPS e número 1 da

lista "Top500" dos computadores mais rápidos do mundo em novembro de 1999. A comparação de desempenho foi baseada em medições de execuções de *benchmarks* e de aplicações científicas. O enfoque principal da análise de desempenho foi quanto à escalabilidade, ou seja, a capacidade de acompanhar o crescimento do tamanho do problema com um crescimento proporcional no poder de processamento. Foi apontada como maior limitação do *cluster* sua grande latência na passagem de mensagens.

HAWICK (2000) estuda o desempenho de um *cluster* tipo Beowulf com 116 nós conectados por rede Fast Ethernet, construído especificamente para pesquisa em Química Computacional na Universidade de Adelaide, Austrália. Como o *cluster* foi projetado para ser utilizado por vários usuários submetendo seus *jobs* simultaneamente, o objetivo era maximizar a vazão (*throughput*) do recurso computacional com um todo e não apenas o seu desempenho para um único *job*. Descreve requisitos para o sistema e as decisões de projeto para a seleção dos componentes do *cluster*, levando em conta o preço e o desempenho que pode ser obtido. Apresenta os resultados de alguns *benchmarks* e faz uma análise comparativa com as plataformas tradicionais para o processamento deste tipo de aplicação.

HAAS (2000) apresenta resultados de experimentos com o modelo ARPS realizados em 3 plataformas paralelas: o IBM SP2 da UFSC; um *cluster* de estações de trabalho do CLIMERH; e o *cluster* Linux do NURCAD/UFSC. Foram utilizados conjuntos de dados meteorológicos de dois casos reais como entrada para o modelo: a) a tempestade de 20 de maio de 1977 em Del City – Oklahoma; e b) a chuva orográfica de 23 de dezembro de 1995 em Florianópolis e Sul do Estado de Santa Catarina. O primeiro caso permitiu avaliar comparativamente a escalabilidade das diferentes plataformas entre si e confrontar os resultados com os de outros experimentos conhecidos na literatura. O segundo caso forneceu evidências da boa estabilidade do modelo e da sua precisão em prever a localização e o tempo em que ocorrerão as precipitações.

De modo geral, estes trabalhos posicionam o *cluster* como uma arquitetura extremamente promissora para o processamento de alto desempenho, quanto a custo e capacidade de processamento. Como diferenças em relação ao trabalho proposto aqui, nenhum dos trabalhos apresentados buscou encontrar o ponto ótimo de custo/benefício

para a construção do *cluster*. Também nenhum dos trabalhos citados desenvolveu modelos para análise do sistema computacional usando uma ferramenta de simulação.

3. Materiais e Métodos

3.1 Introdução

Neste capítulo estão relacionados materiais, métodos e técnicas que fundamentaram o trabalho experimental e a modelagem do *cluster*.

3.2 Metodologia

Para o desenvolvimento deste estudo, foi seguida a metodologia descrita pelas seguintes etapas:

a) **Determinação das Equações do Modelo Analítico** – O desempenho do *cluster* na execução seqüencial e paralela do ARPS foi analisado quanto ao uso de ciclos da CPU e da largura de banda da rede. Foram estabelecidas as equações do modelo analítico do *cluster* para os casos de decomposição unidimensional e bidimensional do domínio;

b) **Modelagem da Carga Imposta pelo ARPS** – Foi criado um modelo de simulação considerando o ARPS como uma carga para o *cluster*, determinada pelo tamanho das mensagens processadas pelos nós e transmitidas através da rede;

c) **Modelagem dos Nós** – Foi criado um modelo de simulação de um nó do *cluster*, considerando os fatores que afetam o desempenho. Este modelo foi reproduzido 64 vezes, pois era o número de nós máximo que se pretendia simular;

d) **Modelagem da Rede –** Foi desenvolvido um modelo de simulação da rede Ethernet comutada, válido para redes de 10, 100 e 1000 Mbps mediante parametrização. O nível de detalhamento foi até a camada de enlace, para levar em conta o efeito do congestionamento na rede quando ela é submetida a uma carga alta;

e) **Experimentação no *Cluster* do NURCAD–** Foram feitos experimentos no *cluster* do NURCAD para levantar amostras de controle que possibilitassem a validação dos modelos de simulação e analítico. Havia 6 nós disponíveis para estes experimentos;

f) **Obtenção das Previsões para o *Cluster* do NURCAD** – Os modelos de simulação e analítico foram executados com a mesma configuração do *cluster* do NURCAD e foram obtidas as previsões de desempenho;

g) **Validação** – Os resultados do item f) foram comparados com as amostras de controle obtidas nas medições no NURCAD levantadas no item e). Os modelos de desempenho foram validados para até 6 nós;

h) **Obtenção das Previsões para Casos Reais de Aplicação do ARPS –** Os modelos validados foram configurados conforme casos reais de aplicação do *cluster* e executados para produzirem suas previsões. Foram escolhidos 3 fatores de interesse para estudo da influência sobre o desempenho: Número de Nós, *Clock* da CPU e Vazão da Rede. O primeiro fator foi variado em 3 níveis e os outros dois foram variados em 2 níveis, totalizando 12 configurações diferentes. O tamanho da grade e a resolução utilizados foram correspondentes a uma previsão do tempo real em Santa Catarina;

i) **Comparação entre as Previsões** – As previsões produzidas no item h) foram comparadas entre si, validando parcialmente ambos os modelos de desempenho. A validação completa só poderia ocorrer a partir de medições de um *cluster* com a mesma configuração;

j) **Análise dos Resultados** – Os resultados das previsões foram analisados para o conjunto de configurações estudadas, sendo destacadas aquelas que cumpriram a meta de tempo de execução do ARPS, estabelecida em 1 h. 30 min. para este trabalho. A importância relativa dos 3 fatores na variação observada nos resultados da simulação foi avaliada, indicando quais os fatores que poderiam ser objeto de um estudo experimental mais detalhado.

3.3 Ferramentas para Simulação

O uso da simulação muitas vezes traz vantagens em relação ao uso de modelos analíticos de desempenho (JAIN, 1991). Em particular, para a avaliação do desempenho do ARPS, a simulação destaca-se em alguns aspectos:

- Melhora a precisão dos resultados, pois evita o uso de muitas das simplificações e suposições dos modelos analíticos. Por exemplo, pode-se levar em conta o efeito das bordas do *cluster*, onde há menor quantidade de dados transferidos;
- Possibilita a obtenção e análise de qualquer medida de desempenho do sistema que se deseje;
- Facilita a experimentação e avaliação de novas formas de operação do sistema.

A ferramenta de simulação utilizada foi o software Arena, versão 3.01, da Systems Modeling Corporation. A máquina utilizada na execução das simulações foi um PC AMD 500 MHz, com 64 M RAM.

O tempo necessário para simular completamente uma única execução do ARPS no Arena foi de até 13 horas.

3.4 Configuração do *Cluster* do NURCAD

O *cluster* utilizado para os experimentos e validação dos modelos deste trabalho é uma sub-rede da UFSC localizada no Laboratório do NURCAD, representado na Figura 11.

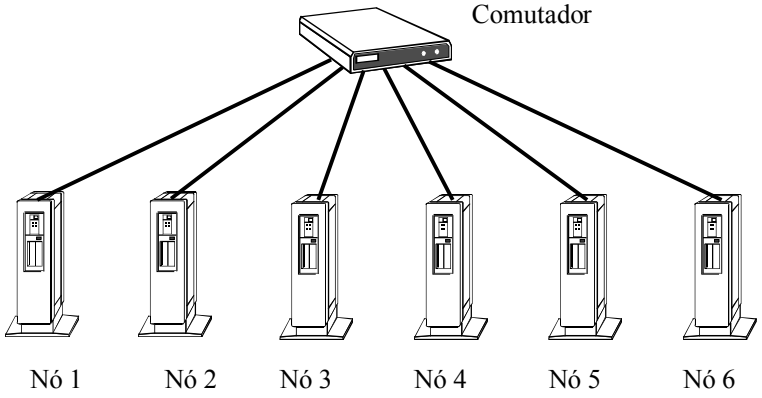


Figura 11 – *Cluster* do NURCAD

A configuração de hardware e software deste *cluster* é a seguinte:

- 6 microcomputadores com CPU Athlon 1 GHz, monoprocessadas, 256 Mbytes de memória RAM, placa de rede 3COM Fast Ethernet 100 Mbps;
- Comutador 3Com, 16 portas, modelo Office Connect Dual Speed Switch 16;
- Sistema Operacional Conectiva Linux 5.0, kernel 2.2.12-20;
- ARPS versão 4.5.1.

3.5 Configuração e Execução do ARPS

A execução do ARPS no *cluster* é configurada em um arquivo de script, através de diversos parâmetros que podem ser ajustados, entre os quais:

- `$nxcpu, $nycpu` : número de nós nas direções X e Y, respectivamente;
- `$nx_glob, $ny_glob, $nz_glob` : número de pontos da grade nas direções X, Y e Z. Devem ser escolhidos de forma que os três parâmetros a seguir sejam números inteiros:

```
$nx = ( $nx_glob -3 ) / $nxcpu + 3;
```

```
$ny = ( $ny_glob -3 ) / $nycpu + 3;

$nz = $nz_glob;
```

- `dx, dy , dz` : Resolução espacial da grade, em metros;
- `tstop` : tempo final de simulação, em segundos;
- `dtbig, dtsml` : passos de tempo maior e menor, em segundos.

A execução do ARPS é iniciada com o comando `arps_mpi`.

3.6 Determinação do Parâmetro t_{unit}

Os modelos analítico e de simulação necessitam do parâmetro t_{unit} para caracterizar o poder de processamento dos nós. Este parâmetro é de natureza empírica e diferente para cada configuração dos nós. Ele pode ser obtido a partir da equação (3) para o caso da execução seqüencial do ARPS:

$$t_{unit} = \frac{T_{seq}}{174.Nx.Ny.Nz + 1574.Nx.Ny.Nz.Nt} \tag{25}$$

Para obter este parâmetro no caso do *cluster* do NURCAD, o ARPS é executado em uma única máquina e mede-se o valor de T_{seq}. O valor de t_{unit} é então calculado e utilizado nos modelos que possuem nós com a mesma configuração.

Para obter t_{unit} de nós com configurações que não estejam disponíveis para medição, utilizamos a seguinte aproximação: extrapolamos linearmente o t_{unit} para o *clock* da CPU disponível, proporcionalmente ao *clock* da CPU desejada. Assim, obtemos um fator de proporcionalidade, que então é multiplicado pelo t_{unit} disponível.

3.7 Determinação dos Parâmetros t_{start} e t_{trans}

As previsões do modelo analítico são calculadas através da equação (24) para T_p, o tempo de execução do código paralelizado para o caso bidimensional. Este tempo é formado por duas parcelas, T_{calc} e T_{itcom}, dadas respectivamente pelas equações (21) e (22). Nesta última equação, são necessários os parâmetros t_{start} e t_{trans} para o cálculo de T_{itcom}.

O parâmetro t_{start} pode ser estimado pela equação (36) para o Retardo de Acesso do modelo de simulação, fazendo-se o número de colisões C = 1:

$$t_{start} = \frac{512}{\text{LargBanda}}$$

O parâmetro t_{trans}, a taxa de transmissão da rede em segundos por palavra, pode ser calculado pela expressão:

$$t_{trans} = \frac{\text{TamPalavra}}{\text{LargBanda}}$$

4. O Modelo de Simulação

4.1 Introdução

O modelo de simulação desenvolvido no Arena inclui as principais características do sistema real que afetam o desempenho do *cluster*. O modelo está pronto para qualquer forma de decomposição, unidimensional ou bidimensional, e assume que o número máximo de nós que participarão da simulação é de 64 nós.

Embora a disposição dos nós no modelo não influa nos resultados, o tamanho da grade foi pré-definido em 8 X 8 nós, para uma melhor visualização das animações que o Arena possibilita. O modelo contém um comutador de 64 portas, dispostas da mesma forma. A tecnologia de redes modelada é a Ethernet, com possibilidade de configurá-la para qualquer vazão de rede: 10, 100 ou 1000 Mbps.

No início da simulação, os blocos de dados são criados e enviados aos nós através da rede, representando os dados transferidos do *host* para os nós durante a inicialização do ARPS. Durante a etapa iterativa, os dados são processados nos nós, gerando novos dados que são transmitidos a seus vizinhos. A simulação termina quando o número de passos de tempo do ARPS houver sido atingido.

4.2 Tamanho das Mensagens

As expressões para o tamanho das mensagens trocadas entre dois nós, utilizadas pelo modelo de simulação, consideram o caso genérico de uma grade retangular, onde os tamanhos das mensagens podem ser diferentes em cada direção.

Seja uma grade com número de pontos N_x, N_y e N_z, decomposta nas direções X e Y, como na Figura 10 do sub-capítulo 2.5.2.2. O número de campos de prognóstico por ponto da grade é 11. Sendo Np o número de pontos da superfície de fronteira entre dois

subdomínios adjacentes e TamPalavra o tamanho da palavra da CPU, o tamanho de uma mensagem iterativa enviada de um subdomínio a outro é:

$$\text{TamMensIt} = 11.\,\text{Np}.\,\text{TamPalavra} \tag{26}$$

Utilizando-se a equação (16) para calcular a superfície da interface na direção X, chega-se à seguinte expressão para o tamanho das mensagens nesta direção:

$$\text{TamMensItX} = \frac{11.\,N_y\,.\,N_z\,.\,\text{TamPalavra}}{P_y} \tag{27}$$

Analogamente, utilizando-se a equação (15) para a direção Y, chega-se a:

$$\text{TamMensItY} = \frac{11.\,N_x\,.\,N_z\,.\,\text{TamPalavra}}{P_x} \tag{28}$$

O tamanho da mensagem de inicialização é deduzido por JOHNSON (1994), a partir da constatação de que, durante a inicialização do ARPS, são enviados para cada nó, por ponto da grade: 7 campos de estado-base; 6 campos de coordenadas, em 2 instantes de tempo; e 11 campos de prognóstico, em 2 instantes de tempo. Isto totaliza 41 valores por ponto da grade. Assim, o tamanho da mensagem de inicialização TamMensBase para cada nó, em bits, é:

$$\text{TamMensBase} = 41.\,\text{TamPalavra}\,.\,\text{NumPontosNo} \tag{29}$$

$$\text{TamMensBase} = \frac{41.N_x.N_y.N_z.\text{TamPalavra}}{P} \tag{30}$$

4.3 Tempo de Execução em Cada Nó

O primeiro fator da equação de T_{calc} (21) é o tempo de computação da fase de inicialização, dado por:

$$T_{init} = 174.N_x.N_y.N_z.t_{unit} \tag{31}$$

O restante da equação (21) representa o tempo de processamento durante todo o tempo de execução do modelo. Reescrevendo para um único passo de tempo ($N_t = 1$):

$$T_{proc} = \frac{\left[1574.N_x.N_y + 1338.\left(N_x.P_y + N_y.P_x\right)\right].N_z}{P}.t_{unit} \quad (32)$$

Estas duas expressões são utilizadas no modelo do de simulação para calcular o retardo de processamento em cada nó, conforme a etapa em execução.

4.4 Caracterização da Rede

Para a modelagem da rede Ethernet, partimos de TANENBAUM (1997), que apresenta uma análise aproximada do desempenho do protocolo CSMA/CD sob condições de carga alta e constante. Considerando k como o número de estações tentando transmitir, a probabilidade média A de que alguma estação adquira o acesso ao meio é dada por:

$$A = \left(\frac{k-1}{k}\right)^{k-1}$$

$$A = \left(1 - \frac{1}{k}\right)^{k-1} \quad (33)$$

Para um número grande de estações (acima de 5), pode-se demonstrar que $A \rightarrow 1/e$ rapidamente. O número médio de *slots* necessários para que ocorra uma transmissão com sucesso é igual a 1/A. Assim, o número de colisões que ocorrem antes de um sucesso ocorrer é de:

$$C = \frac{1}{A} - 1$$

$$C = \frac{1}{\left(1 - \frac{1}{k}\right)^{k-1}} - 1 \quad (34)$$

Cada *slot* de tempo tem uma duração S, igual ao tempo de propagação do quadro mínimo de 512 bits (64 bytes) por uma rede com largura de banda LargBanda:

$$S = TempoPropag = \frac{512}{LargBanda} \tag{35}$$

O retardo que uma mensagem sofre ao ser transferida pela rede pode ser definido com base nos seguintes conceitos (SOARES, 1995):

a) Retardo de Acesso – É o intervalo decorrido desde que uma mensagem a transmitir é gerada pela estação até o momento em que a estação consiga obter para ela e somente para ela o direito de transmitir, sem que haja colisão de mensagens no meio;

b) Retardo de Transmissão – É o intervalo de tempo decorrido desde o início da transmissão de uma mensagem por uma estação de origem até o momento em que a mensagem chega à estação de destino;

c) Retardo de Transferência – É a soma do Retardo de Acesso com o Retardo de Transmissão. Inclui todo o tempo de entrega de uma mensagem, desde o momento em que se deseja transmiti-la, até o momento em que ela chega para ser recebida pelo destinatário.

O Retardo de Acesso é expresso pelo tempo perdido com uma colisão multiplicado pelo número de colisões ocorridas, portanto:

$$DelayAcesso = S \,.\, C \tag{36}$$

$$DelayAcesso = TempoPropag. \left[\frac{1}{\left(1 - \frac{1}{P}\right)^{P-1}} - 1 \right] \tag{37}$$

O Retardo de Transmissão de uma rede a partir do momento em que o meio está à disposição, é dado por:

$$DelayTransm = \frac{TamQuadro}{LargBanda} \tag{38}$$

O Retardo de Transferência médio é a soma dos retardos das equações (37) e (38):

DelayRede = DelayAcesso + DelayTransm

$$\text{DelayRede} = \text{TempoPropag} \cdot \left(\frac{1}{\left(1 - \frac{1}{P}\right)^{P-1}} - 1 \right) + \frac{\text{TamQuadro}}{\text{LargBanda}} \qquad (39)$$

Ao entrar na rede, o pacote é encapsulado em quadros de até 1500 bytes. Cada quadro entra em uma fila em ordem aleatória. Quando o quadro ganha acesso ao meio e é transmitido, sofre um retardo igual a DelayRede. Os quadros vão sendo remontados à medida que são transmitidos, até completar o pacote, quando então este é enviado ao nó de destino.

O número de quadros em que um pacote de tamanho TamPacote será encapsulado é calculado pela equação:

$$\text{NumQuadrosTot} = \text{INT}\left(\frac{\text{TamPacote}}{12000} \right) + 1, \qquad (40)$$

onde INT(A) indica a parte inteira de A.

4.5 Coordenadas dos Nós e Endereçamento

No modelo considera-se que o número de nós pode variar de 1 a 64, através da configuração dos parâmetros P_x e P_y. Cada nó possui um número de identificação dest, variando de 1 a 64, e está conectado à porta de mesmo número no comutador. O identificador dest é utilizado como índice dos vetores Nos( ) e Comutadores( ), permitindo localizar um nó ou uma porta do comutador no momento de endereçar os pacotes de dados.

Durante a etapa de inicialização, é enviada uma mensagem para cada nó. Para determinar quais os nós e portas do comutador que participarão da simulação de uma malha com número de nós $P = P_x \cdot P_y$, faz-se a variável indice assumir valores entre 1 e P na equação:

$$dest = MOD(indice - 1, P_x) + 1 + MaxP_x * INT\left(\frac{indice - 1}{P_x}\right) \quad (41)$$

onde $MaxP_x$ é o número máximo de nós na direção X, igual a 8.

Por exemplo, para uma malha de 12 nós, com $P_x = 4$ e $P_y = 3$, o identificador dest seria atribuído aos nós conforme a Figura 12. Somente os nós com estes números de identificação participariam da simulação.

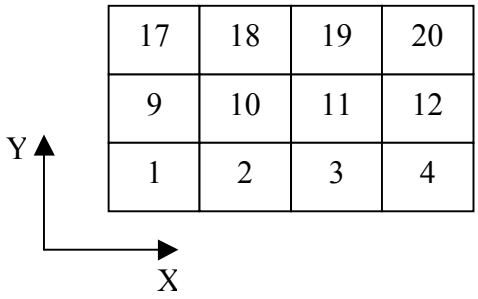


Figura 12 – Identificação dos subdomínios para uma grade 4 X 3

Como a lógica de endereçamento baseia-se sempre nas coordenadas X e Y dos nós, durante a etapa de inicialização as coordenadas são calculadas pelas seguintes equações, a partir do identificador dest do nó:

$$IDNoX = MOD((dest - 1), P_x) + 1 \quad (42)$$

e

$$IDNoY = INT\left(\frac{dest - 1}{P_x}\right) + 1, \quad (43)$$

onde MOD(A,B) indica o resto da divisão de A por B. Para o exemplo da Figura 12, IDNoX assumiria valores de 1 a 4 e IDNoY assumiria valores de 1 a 3.

Quando uma mensagem precisa ser endereçada aos nós adjacentes, o modelo de simulação calcula o número de identificação de todos os vizinhos existentes. Este cálculo deve ser suficientemente genérico para qualquer configuração da grade e posição do nó. Sendo dest o identificador do nó atual, com coordenadas IDNoX e IDNoY, a existência e a identificação dos nós de destino são determinadas através das seguintes condições e equações:

- Se IDNoX > 1 , então dest1 = dest – 1 ;
- Se IDNoX < P_x , então dest2 = dest + 1 ;
- Se IDNoY > 1 , então dest3 = dest – $MaxP_x$;
- Se IDNoY < P_y , então dest4 = dest + $MaxP_x$.

Um nó aguarda a chegada dos resultados de todos os nós vizinhos antes de iniciar o processamento do próximo passo de tempo. O número de nós vizinhos é determinado pelo número de condições acima que resultarem verdadeiras, o que pode ser obtido no Arena somando-se o resultado lógico (1 ou 0) das expressões de comparação:

$$\begin{aligned} NumVizinhos = (IDNoX > 1) + (IDNoX < Px) + \\ + (IDNoY > 1) + (IDNoY < Py) \end{aligned} \quad (44)$$

4.6 Controle da Simulação

O controle da duração da simulação é efetuado através da variável PassosProcessados, que conta o número de entidades processadas no nó número 1. Quando PassosProcessados for igual a N_t (o número de passos de tempo de uma rodada de execução do ARPS), a simulação é encerrada.

4.7 Percurso de uma Entidade pelo Sistema

No modelo de simulação desenvolvido, as entidades movimentadas pelo sistema são os blocos de dados (quadros, pacotes). O processamento dá-se em dois estágios: nos nós e na rede. A rede recebe e encaminha todos os blocos de dados que precisam ser transferidos de um nó de origem a um nó de destino. O percurso de uma entidade pelo modelo pode ser melhor compreendido acompanhando-se o fluxograma de execução do modelo de simulação, representado na Figura 13.

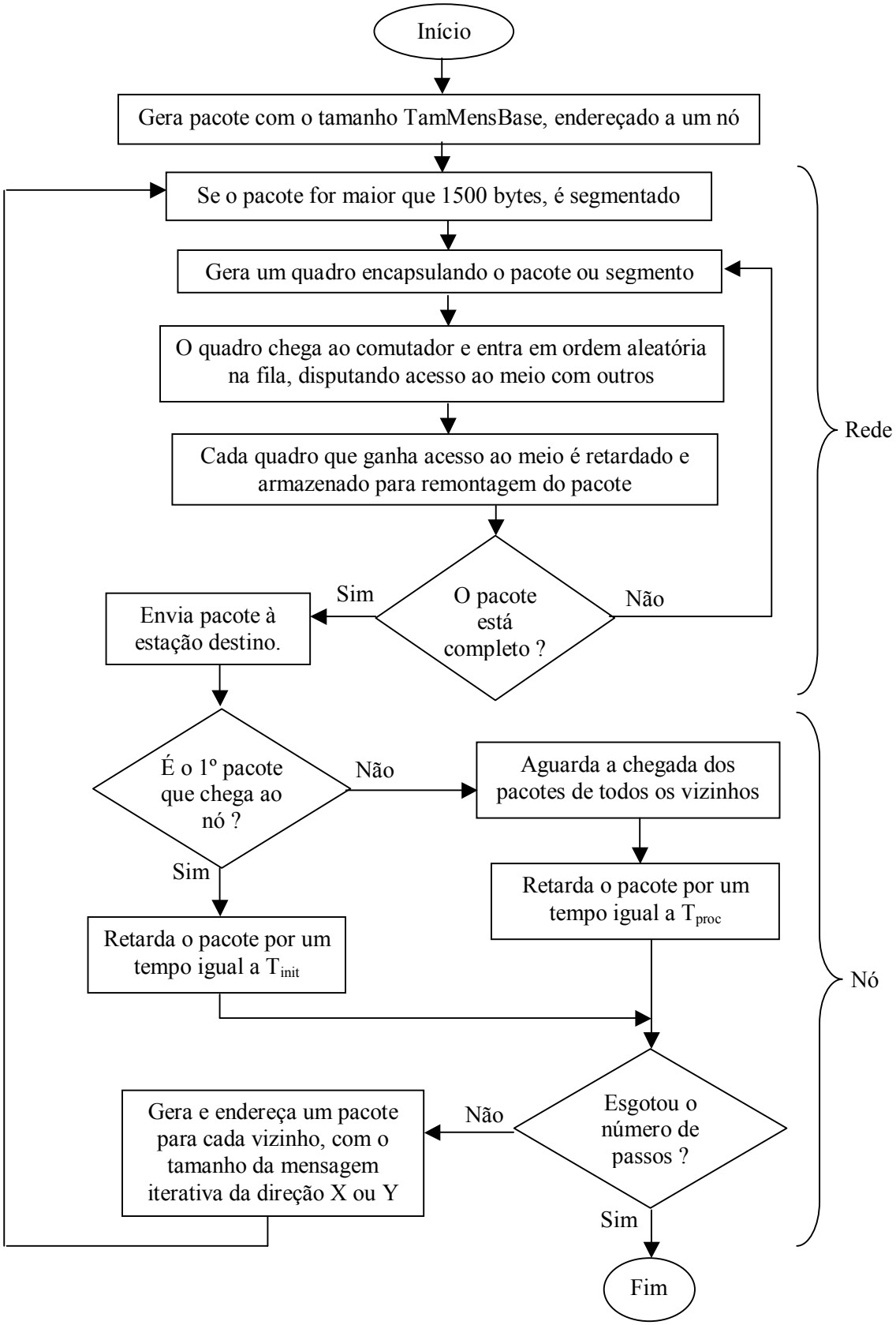


Figura 13– Fluxograma de execução do modelo de simulação

A partir daqui, seguiremos o caminho que uma entidade percorre pelo modelo de simulação, analisando as funções desempenhadas em cada estágio.

A simulação inicia com a criação de pacotes de dados, um para cada nó da grade, representando os dados transferidos do *host* para os nós durante a inicialização do ARPS. Isto é realizado no módulo Arrive da Figura 14. Os pacotes gerados têm tamanho igual ao da mensagem de inicialização, TamMensBase, e possuem um número de identificação único. Também recebem o endereço do seu nó de destino, definido como um identificador dest, a partir do qual são calculadas as coordenadas IDNoX e IDNoY.

A seguir, os pacotes chegam à rede e são encapsulados em quadros Ethernet. Caso o tamanho do pacote supere o MTU da rede (1500 bytes), ocorre o seu encapsulamento em mais de um quadro. O número total de quadros gerados para o pacote é armazenado no atributo NumQuadrosTot. Então cada quadro é encaminhado à porta do comutador onde está conectado o seu nó de destino.

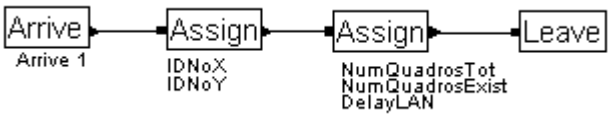


Figura 14 – Criação dos pacotes na etapa de inicialização

Apenas um quadro de cada pacote é transmitido por vez pela rede; os demais quadros do mesmo pacote, se existirem, serão gerados somente quando o anterior já tiver saído do comutador. Caso já exista algum quadro sendo transmitido para esta porta do comutador, o quadro entra em uma fila, em posição aleatória. A chegada da vez de um quadro nesta fila representa a obtenção do acesso ao meio de transmissão por parte do nó que originou o quadro. Isto simula um retardo não-determinístico, que é uma característica das redes Ethernet. O retardo que cada quadro sofre ao ser processado é dado pela soma do Retardo de Acesso e do Retardo de Transmissão. O fragmento do modelo apresentado na Figura 15 mostra 6 portas do comutador.

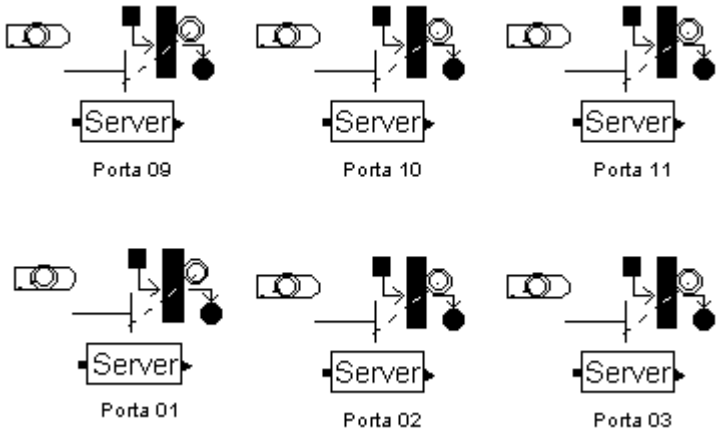


Figura 15 – Seção do comutador

À medida que os quadros são transmitidos pelo comutador, vão sendo armazenados para futura remontagem do pacote. O número de quadros já transmitidos é contabilizado pela variável NumQuadrosExist. Quando todos os quadros que pertencem a um mesmo pacote, verificado através do número de identificação único, passam através da rede, o pacote original é remontado e remetido ao nó de destino. Esta etapa é modelada no fragmento do modelo representado na Figura 16.

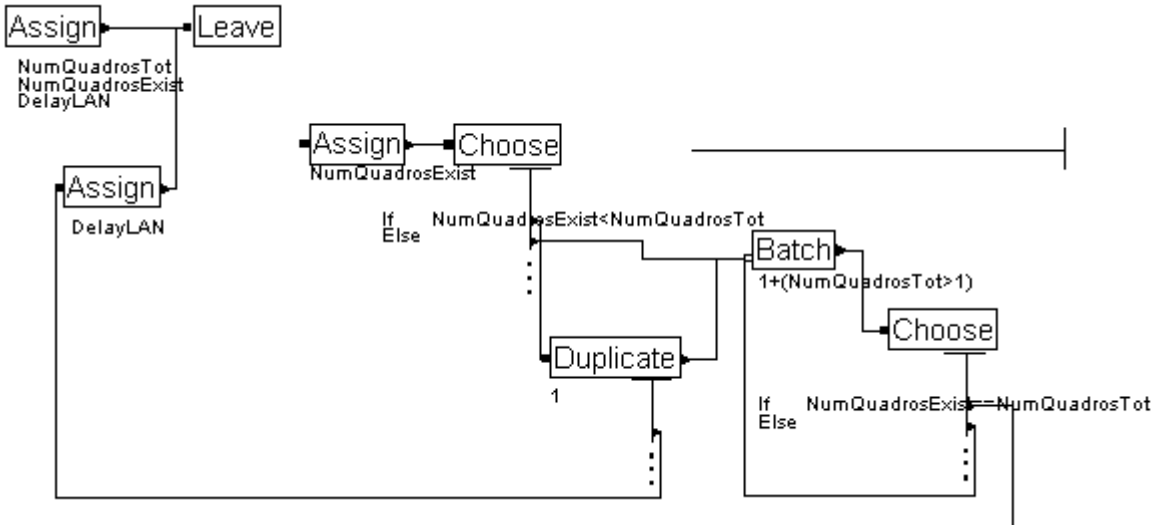


Figura 16 – Lógica de encapsulamento e remontagem dos pacotes

Ao chegar ao nó, cada pacote fica armazenado até que sejam recebidos os pacotes de todos os nós vizinhos, número que pode variar de 1 a 4 conforme a

localização do nó e o tipo de decomposição da grade. Somente quando chegarem todos os pacotes é que o nó estará pronto para iniciar o processamento dos dados. Ver fragmento do modelo representado na Figura 17.

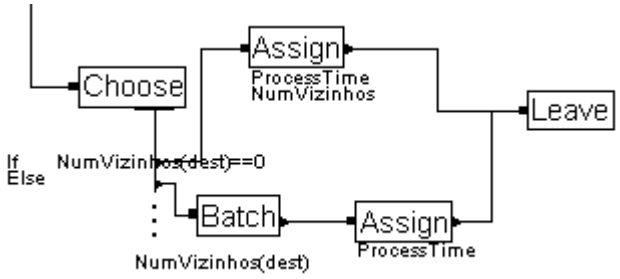


Figura 17 – Controle de chegada de dados nos nós

O tempo de processamento em um nó é uma função do número de operações de ponto flutuante executadas pelo ARPS em cada etapa, podendo assumir dois valores: T_{init} durante a etapa de inicialização, ou T_{proc} durante a etapa iterativa. T_{init} é o tempo de processamento utilizado somente para o primeiro pacote. T_{proc}, utilizado para todos os demais pacotes, foi mantido constante em todas as simulações executadas neste trabalho; porém o modelo está preparado para aceitar qualquer distribuição de probabilidade que represente a variabilidade do tempo de execução do ARPS nos nós quando é processado um modelo real. Ver fragmento do modelo representado na Figura 18.

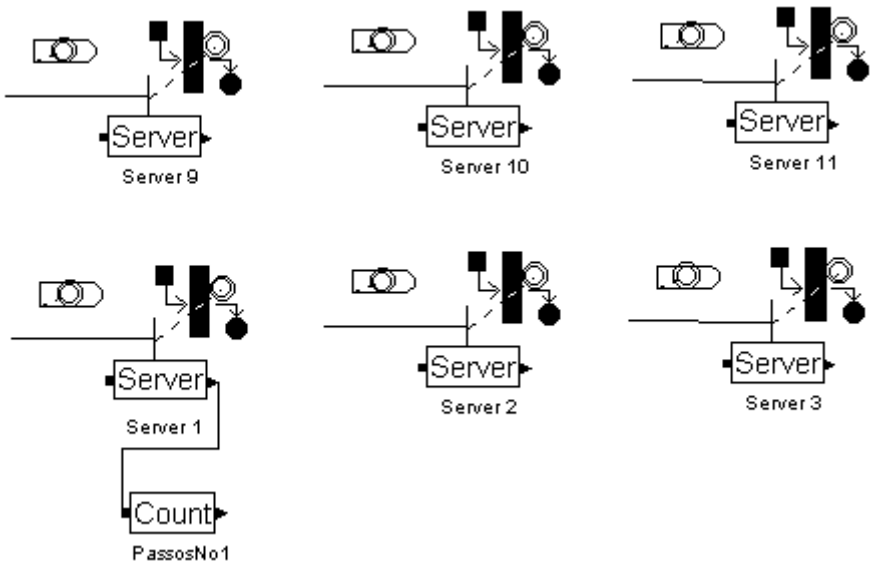


Figura 18 – Seção da grade

Quando um nó termina de processar seus dados, estes resultados devem ser enviados a todos seus vizinhos. O modelo gera de 1 a 4 pacotes de dados, conforme a localização do nó e o tipo de decomposição da grade, cada um endereçado para um nó vizinho. Cada pacote recebe um número de identificação único. A ele também é atribuído um tamanho, de acordo com a quantidade de dados que devem ser transferidos para o nó vizinho: TamMensItX se a transferência for para um vizinho na direção X, ou TamMensItX se a transferência for para um vizinho na direção Y. Esta etapa é modelada no fragmento do modelo representado na Figura 19.

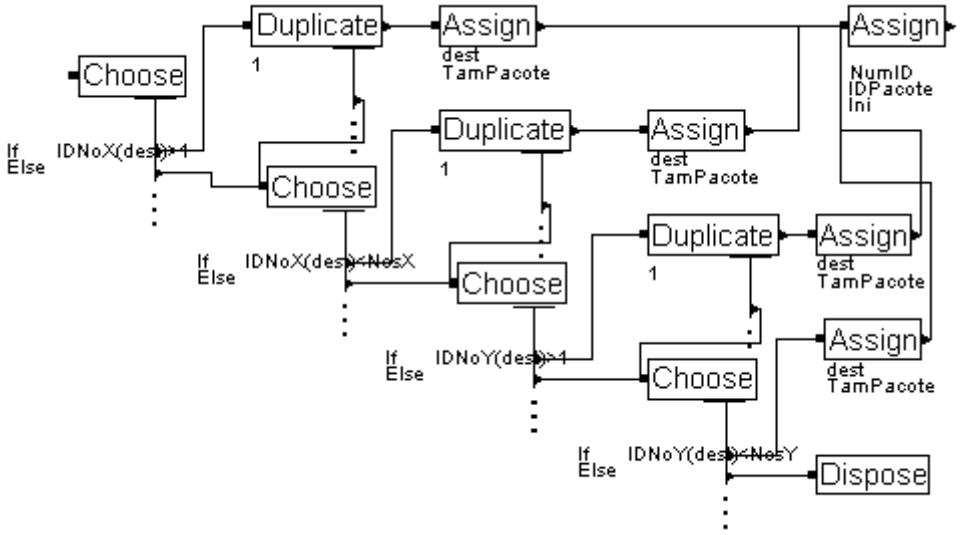


Figura 19 – Criação e endereçamento de pacotes na saída de um nó

A simulação é encerrada quando o número de pacotes processados nos nós atinge o número de passos de tempo estipulado no modelo ARPS.

Na Figura 20 está representada uma visão geral de todo o modelo de simulação no Arena: na parte de cima, estão dispostos os 64 nós da grade; no centro, à direita, estão localizadas as 64 portas do comutador; no centro, à esquerda, aparece a lógica de endereçamento dos pacotes; e abaixo, à esquerda, figuram os módulos referentes à rede.

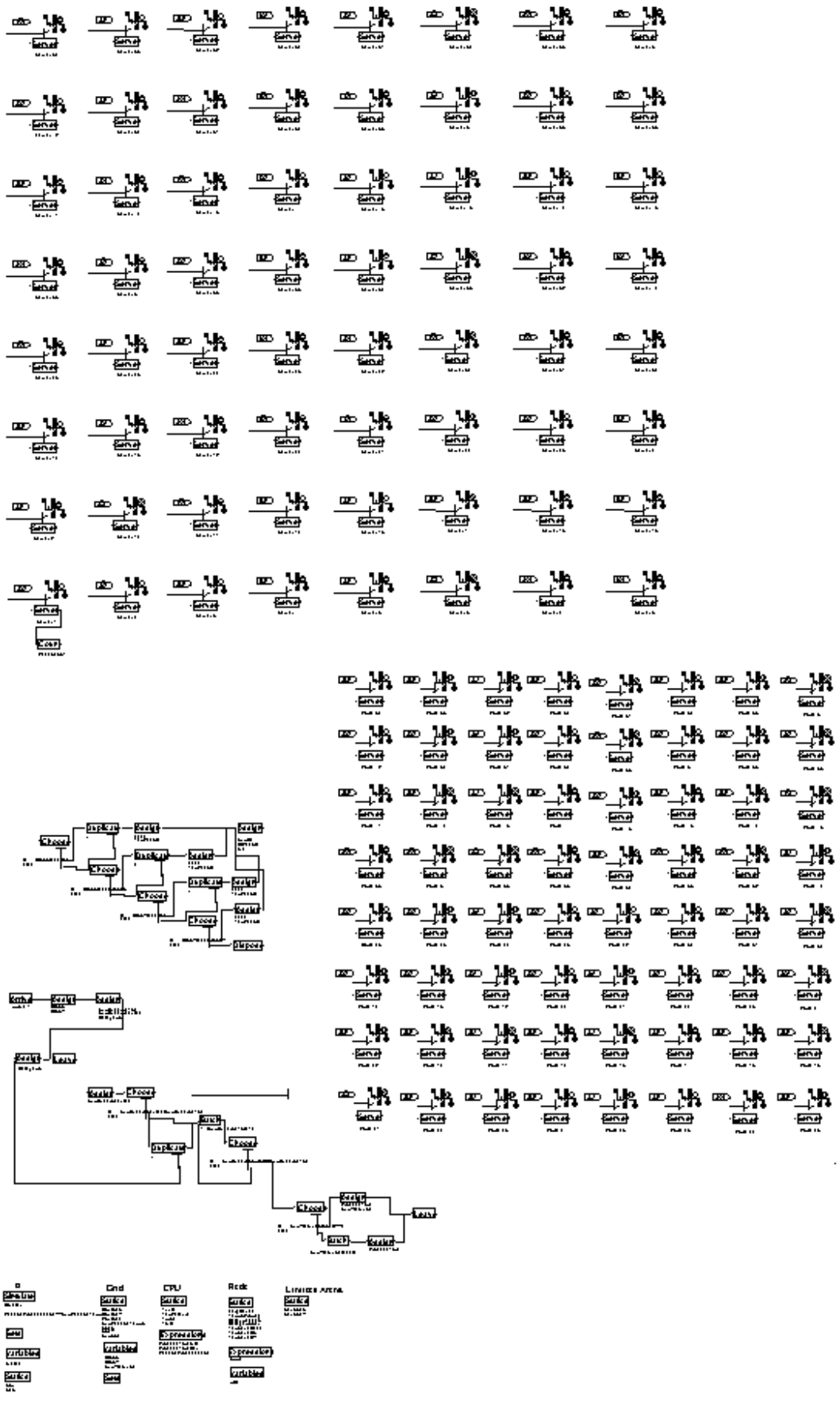


Figura 20 – Visão geral do modelo de simulação no Arena

4.8 Dicionário de Dados do Modelo

A seguir estão relacionadas as constantes, variáveis, atributos e expressões que definem a configuração do modelo para uma simulação.

- **Referentes à grade:**

- P_x, P_y : número de nós (ou processadores) nas direções X e Y, respectivamente;

- N_x, N_y, N_z : número de pontos da grade nas direções X, Y e Z, respectivamente;

- NumVizinhos : número de vizinhos de um determinado nó;

- $MaxP_x$, $MaxP_y$: número máximo de nós (ou processadores) dispostos nas direções X e Y, respectivamente, para melhor visualização das animações; mantido constante com valor 8.

- **Referentes ao processamento em um nó:**

- t_{unit} : tempo típico necessário para o processamento de uma operação de ponto flutuante genérica, em segundos;

- T_{proc} : tempo para executar todos os cálculos de um passo do modelo ARPS, em segundos;

- T_{init} : tempo para configurar o nó com os dados de inicialização, em segundos;

- TamPalavra : tamanho da palavra da CPU, em bits.

- **Referentes à rede:**

- LargBanda : taxa de bits nominal da rede, em bps;

- TempoPropag: duração de um *slot* de tempo, correspondente à transmissão de um quadro de 512 bits (64 bytes), em segundos;

- DelayAcesso : tempo médio que decorre até que uma estação consiga o direito de transmitir no meio compartilhado, em segundos;

- DelayRede: retardo de transferência de um quadro, em segundos;

- NumQuadrosTot : número total de quadros em que um pacote foi encapsulado;

- NumQuadrosExist : número de quadros de um pacote que já foram transmitidos.

- **Referentes às mensagens transmitidas entre os nós:**

- TamMensBase : tamanho da mensagem de inicialização transmitida a todos os nós, em bits;

- TamMensItX : tamanho das mensagens iterativas transmitidas entre subdomínios na direção X, em bits;

- TamMensItY : tamanho das mensagens iterativas transmitidas entre subdomínios na direção Y, em bits.

- **Referentes ao Controle da Simulação**

- N_t : número de passos de tempo de uma rodada de execução do ARPS, em segundos;

- PassosProcessados : número de entidades processadas no nó número 1.

5. Resultados Experimentais

5.1 Introdução

Neste capítulo serão primeiro apresentados os dados obtidos com experimentos no *cluster* do NURCAD. Estes dados são de grande importância, pois refletem um ambiente de execução do ARPS plenamente funcional, apenas em menor escala em comparação ao *cluster* que se pretende construir, permitindo validar os modelos analítico e de simulação.

Aqui também serão apresentados os resultados das previsões dos modelos analítico e de simulação para 12 configurações de interesse do *cluster*. Embora seja necessário um estudo com um número maior de experimentos para dar exatidão às conclusões, estes resultados já fornecem uma indicação da influência dos fatores estudados no desempenho do *cluster*. Além disso, as simulações destas configurações são uma demonstração da capacidade preditiva do modelo de simulação desenvolvido.

5.2 Experimentos Realizados no *Cluster* do NURCAD

Os experimentos conduzidos no *cluster* do NURCAD objetivam o levantamento de dados para validação dos modelos analítico e de simulação. Como hoje este *cluster* possui apenas 6 nós, além deste limite deve-se considerar os modelos como extrapolações. A validação completa dos modelos para um número maior de nós não pode ser obtida com estes experimentos.

5.2.1 Execução Seqüencial do ARPS

Tanto o modelo de simulação quanto o modelo analítico dependem do levantamento do parâmetro empírico t_{unit}. O primeiro experimento realizado no *cluster* foi a execução seqüencial do ARPS em uma única máquina, visando obter o parâmetro t_{unit} dos nós, conforme o método descrito no sub-capítulo 3.6.

O ARPS foi executado com os parâmetros $N_x = 67$, $N_y = 67$, $N_z = 35$ e $N_t = 60$. O tempo de execução T_{seq} medido foi de 280,7 s. Utilizando-se a Equação (25), chegamos a $t_{unit} = 18{,}88\ \eta s$ para os nós do *cluster* do NURCAD.

5.2.2 Execução Paralela do ARPS

Os experimentos de execução paralela do ARPS foram realizados variando-se o número de nós de 1 a 6. O número de pontos N_y precisou ser ajustado nos experimentos com 3, 5 e 6 nós, para que o ARPS sempre trabalhasse com número inteiro de pontos por nó, como visto no sub-capítulo 3.5. Na Tabela 1 são apresentadas as medições nos experimentos paralelos e no experimento seqüencial.

Número de Nós	Tamanho da Grade (N_x, N_y, N_z)	T_P Medido [s]
1	67 X 67 X 35	280,7
2	67 X 67 X 35	146,6
3	67 X 66 X 35	104,5
4	67 X 67 X 35	84,2
5	67 X 68 X 35	70,2
6	67 X 69 X 35	62,3

Tabela 1 – Resultados de medições da execução do ARPS no *cluster*

5.3 Validação dos Modelos de Desempenho

Os modelos analítico e de simulação foram configurados com os mesmos parâmetros utilizados nos experimentos do sub-capítulo 5.2.2; o número de nós foi variado de 2 a 6. Os dados resultantes de cada modelo são mostrados na Tabela 2 e na Tabela 3, em comparação com os resultados das medições no *cluster*. No gráfico da Figura 21 são comparados os resultados de ambos os modelos com as medições, para o caso da execução paralela do ARPS.

Número de Nós	T_P Medido [s] (A)	T_P Simulação [s] (B)	Diferença: (B-A)/A
1	280,7	–	–
2	146,6	145,7	-0,6 %
3	104,5	97,5	-6,7 %
4	84,2	75,4	-10,5 %
5	70,2	62,2	-11,4 %
6	62,3	53,5	-14,1 %

Tabela 2 – Resultados das simulações em comparação com as medições

Número de Nós	T_P Medido [s] (A)	T_P Analítico [s] (B)	Diferença: (B-A)/A
1	280,7	–	–
2	146,6	145,1	-1,0%
3	104,5	97,0	-7,2%
4	84,2	75,1	-10,8%
5	70,2	62,0	-11,7%
6	62,3	53,3	-14,4%

Tabela 3 – Resultados das equações analíticas em comparação com as medições

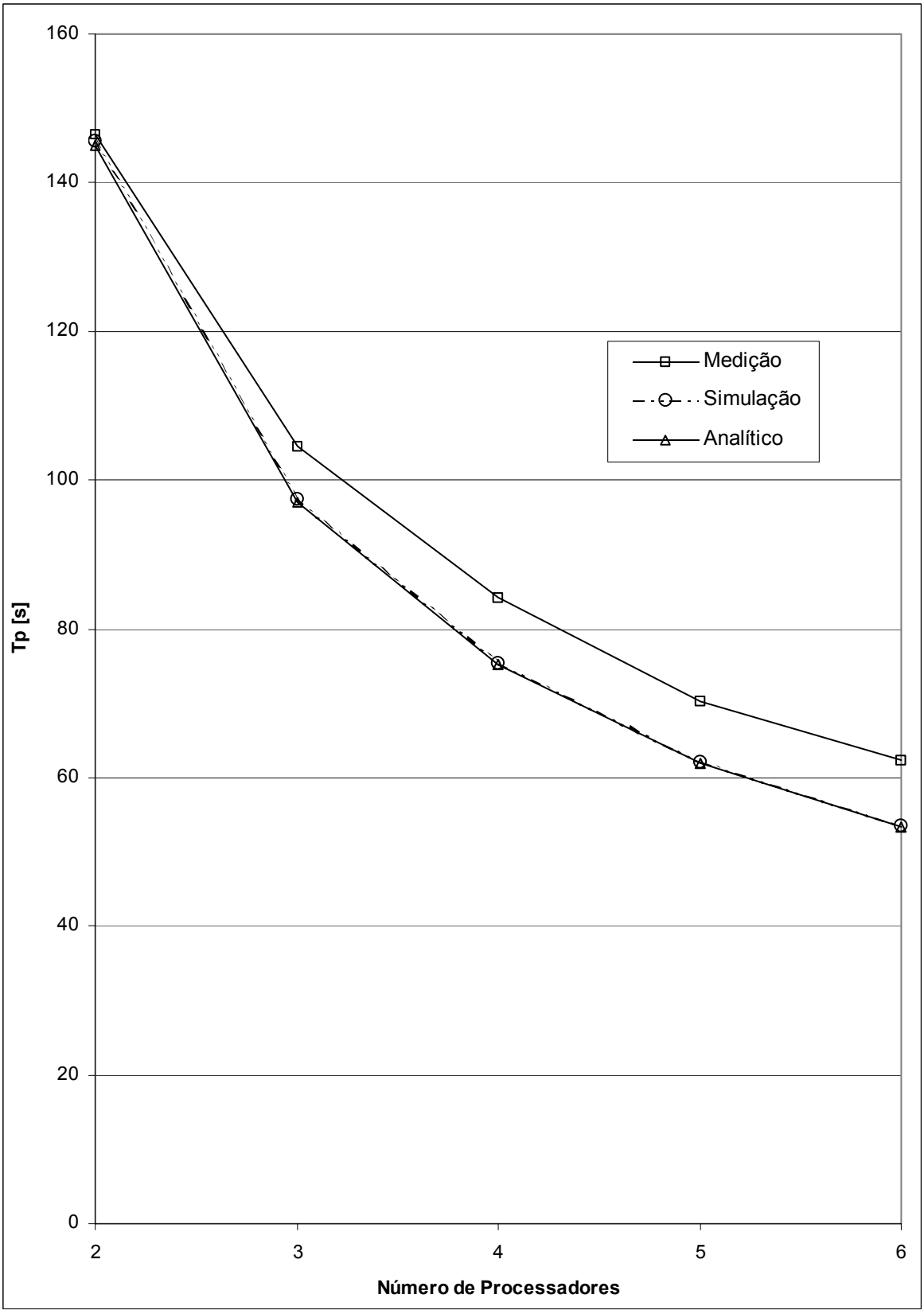


Figura 21 – Gráfico comparando resultados medidos e obtidos dos modelos.

A partir destes dados, pode-se observar que:

- A diferença entre os resultados do modelo de simulação e das medições foi de no máximo 14,1 %;
- A diferença entre os resultados do modelo analítico e das medições foi de no máximo 14,4 %;
- A diferença entre as medições e as previsões dos modelos tende a aumentar à medida que cresce o número de nós;
- Os tempos previstos pelo modelo de simulação ficaram sempre abaixo dos valores medidos e um pouco acima dos tempos previstos pelo modelo analítico.

Os resultados apresentados na Tabela 2 e na Tabela 3 validam os modelos de desempenho para até 6 nós. Para um número de nós maior que este, é possível esperar previsões abaixo dos valores reais de tempo de execução do ARPS, com um erro de no mínimo 14%. Pode-se considerar que os modelos analítico e de simulação fornecem estimativas otimistas do desempenho do *cluster*.

5.4 Previsões de Desempenho em Casos Reais

Utilizando os modelos de desempenho validados no sub-capítulo 5.3, passamos ao estudo de casos reais de aplicação do *cluster*. Para refletir a situação de uma previsão do tempo real em Santa Catarina, foram feitas as seguintes alterações nos parâmetros:

- **Tamanho da Grade**: aumentado para $N_x = 120$, $N_y = 120$ e $N_z = 35$ pontos;
- **Número de Passos de Tempo**: aumentado para $N_t = 17280$, o que corresponde a uma previsão do tempo para 48 h, com $\Delta t = 10s$;
- **Tipo de Decomposição:** alterada para bidimensional, com grade quadrada.

Foram escolhidos 3 fatores de interesse para o estudo da influência sobre o desempenho: o Número de Nós, o *Clock* da CPU e a Vazão da Rede. Os níveis de variação selecionados para estes fatores foram os seguintes:

- **Número de Nós**: 16, 36 e 64;
- ***Clock* da CPU**: 1 GHz e 2 GHz;
- **Vazão da Rede**: 100 e 1000 Mbps.

Para experimentar todas as combinações possíveis de níveis e fatores, são necessários 12 experimentos. Os demais parâmetros foram mantidos fixos durante os experimentos.

Para os experimentos com CPU de 2 GHz é necessário obter-se um novo t_{unit} para os modelos de desempenho. Como esta plataforma não estava disponível no momento da realização deste trabalho, impedindo a medição do t_{unit} em uma execução do ARPS, utilizamos o procedimento descrito no sub-capítulo 3.6. Assim, para a CPU de 2 GHz:

$$t_{unit} = \frac{1\,\text{GHz}}{2\,\text{GHz}}.18{,}88\,\eta s = 9{,}44\,\eta s$$

5.4.1 Previsões do Modelo Analítico

As previsões do modelo analítico foram calculadas através das equação (24) para T_p, e pelas equações (21) e (22) para T_{calc} e T_{itcom}, respectivamente. Os parâmetros t_{start} e t_{trans}, necessários para o cálculo de T_{itcom}, foram determinados segundo o método do sub-capítulo 3.7:

- t_{start} = 5,12 μs para rede de 100 Mbps e t_{start} = 0,512 μs para rede de 1000 Mbps;
- t_{trans} = 320 ηs/palavra para rede de 100 Mbps e t_{trans} = 32 ηs/palavra para rede de 1000 Mbps, considerando-se um tamanho de palavra de 32 bits.

Os resultados das previsões do modelo analítico para o tempo de execução do ARPS estão relacionados na Tabela 4.

Configuração				Tempo de Execução Previsto [s], [hh mm]	
	Número de Nós	*Clock* da CPU [GHz]	Vazão da Rede [Mbps]		
1	16	1	100	18116	5 h 02 min
2	16	1	1000	17196	4 h 47 min
3	16	2	100	9569	2 h 39 min
4	16	2	1000	8649	2 h 24 min
5	36	1	100	8824	2 h 27 min
6	36	1	1000	7904	2 h 12 min
7	36	2	100	4923	1 h 22 min
8	36	2	1000	4003	1 h 07 min
9	64	1	100	5526	1 h 32 min
10	64	1	1000	4606	1 h 17 min
11	64	2	100	3274	55 min
12	64	2	1000	2354	39 min

Tabela 4 – Previsões do modelo analítico

A Tabela 5 mostra a composição de T_p em termos de T_{calc} e T_{itcom}.

Configuração		(Modelo Analítico)	T_{calc}		T_{itcom}	
	[CPU/GHz/Mbps]		[s]	T_{calc}/T_p %	[s]	T_{itcom}/T_p %
1	16/1/100	18116	17094	94,4%	1022	5,6%
2	16/1/1000	17196	17094	99,4%	102	0,6%
3	16/2/100	9569	8547	89,3%	1022	10,7%
4	16/2/1000	8649	8547	98,8%	102	1,2%
5	36/1/100	8824	7802	88,4%	1022	11,6%
6	36/1/1000	7904	7802	98,7%	102	1,3%
7	36/2/100	4923	3901	79,2%	1022	20,8%
8	36/2/1000	4003	3901	97,5%	102	2,5%
9	64/1/100	5526	4504	81,5%	1022	18,5%
10	64/1/1000	4606	4504	97,8%	102	2,2%
11	64/2/100	3274	2252	68,8%	1022	31,2%
12	64/2/1000	2354	2252	95,7%	102	4,3%

Tabela 5 – Composição do T_p no modelo analítico

5.4.2 Previsões do Modelo de Simulação

A configuração e a execução específicas do modelo de simulação são realizadas conforme descrito no capítulo 4. Os resultados obtidos para o tempo de execução do ARPS estão relacionados na Tabela 6.

Configuração				Tempo de Execução Simulado	
	Número de Nós	*Clock* da CPU [GHz]	Vazão da Rede [Mbps]	[s]	[hh mm]
1	16	1	100	17265	4 h 48 min
2	16	1	1000	17111	4 h 45 min
3	16	2	100	8718	2 h 25 min
4	16	2	1000	8564	2 h 23 min
5	36	1	100	7916	2 h 12 min
6	36	1	1000	7813	2 h 10 min
7	36	2	100	4015	1 h 07 min
8	36	2	1000	3912	1 h 05 min
9	64	1	100	4589	1 h 16 min
10	64	1	1000	4512	1 h 15 min
11	64	2	100	2338	39 min
12	64	2	1000	2261	38 min

Tabela 6 – Previsões do modelo de simulação

Já que o modelo de simulação desenvolvido no Arena permite a monitoração individual dos recursos, podemos obter os dados de utilização de cada nó e de cada porta do comutador. Como exemplo dos dados que estão disponíveis, selecionamos a configuração número 7 (6X6 nós, 2GHz e 100Mbps), mostrando na Tabela 7 como a utilização de cada porta varia em função das coordenadas X e Y. Os dados de utilização média dos nós e das portas do comutador são apresentados para todas as configurações simuladas na Tabela 8.

Y	6	2,3%	3,4%	3,4%	3,4%	3,4%	2,3%
	5	3,4%	4,6%	4,6%	4,6%	4,6%	3,4%
	4	3,4%	4,6%	4,6%	4,6%	4,6%	3,4%
	3	3,4%	4,6%	4,6%	4,6%	4,6%	3,4%
	2	3,4%	4,6%	4,6%	4,6%	4,6%	3,4%
	1	2,3%	3,4%	3,4%	3,4%	3,4%	2,3%
		1	2	3	4	5	6
		X					

Tabela 7 – Utilização das portas do comutador.

Exemplo para a simulação com 6X6 nós, 2GHz e 100Mbps, mostrando como a utilização de cada porta varia em função das coordenadas X e Y.

Configuração		Simulação [s]	Utilização Média Comutador	Utilização Média? Nós
	[CPU/GHz/Mbps]			
1	16/1/100	17265	1,2%	99,0%
2	16/1/1000	17111	0,1%	99,9%
3	16/2/100	8718	2,4%	98,0%
4	16/2/1000	8564	0,2%	99,8%
5	36/1/100	7916	2,0%	98,6%
6	36/1/1000	7813	0,2%	99,9%
7	36/2/100	4015	3,8%	97,2%
8	36/2/1000	3912	0,4%	99,7%
9	64/1/100	4589	2,6%	98,1%
10	64/1/1000	4512	0,3%	99,8%
11	64/2/100	2338	5,1%	96,3%
12	64/2/1000	2261	0,5%	99,6%

Tabela 8 – Utilização do comutador e dos nós

5.4.3 Comparação entre as Previsões dos Modelos

Uma comparação entre as previsões produzidas pelos modelo analítico e de simulação é apresentada na Tabela 9.

Configuração		Modelo Analítico [s] (A)	Modelo Simulação [s] (B)	Diferença: (B-A)/A
	[CPU/GHz/Mbps]			
1	16/1/100	18116	17265	-4,7%
2	16/1/1000	17196	17111	-0,5%
3	16/2/100	9569	8718	-8,9%
4	16/2/1000	8649	8564	-1,0%
5	36/1/100	8824	7916	-10,3%
6	36/1/1000	7904	7813	-1,2%
7	36/2/100	4923	4015	-18,4%
8	36/2/1000	4003	3912	-2,3%
9	64/1/100	5526	4589	-17,0%
10	64/1/1000	4606	4512	-2,0%
11	64/2/100	3274	2338	-28,6%
12	64/2/1000	2354	2261	-4,0%

Tabela 9 – Comparação entre as previsões dos dois modelos

De modo geral, os dois modelos produziram previsões bastante similares, com o módulo da diferença ficando abaixo de 20%, à exceção de uma única configuração. Isto valida parcialmente ambos os modelos, restando somente executar o ARPS em um *cluster* com estas mesmas configurações para obter-se medidas reais que possibilitariam a validação completa.

Ainda na Tabela 9, observa-se que o módulo da diferença entre os resultados dos dois modelos é maior (de 4,7 a 28,6%) para as configurações com rede 100 Mbps, e é menor (de 0,5 a 4,0%) para as configurações com rede de 100 Mbps. Isto pode ser atribuído ao fato do modelo analítico não levar em conta a separação dos domínios de colisão proporcionada por cada porta do comutador, resultando em um Retardo de

Transferência superestimado para os quadros da rede e em uma tendência prematura de congestionamento nas configurações com rede de menor vazão (100 Mbps). Como o modelo de simulação leva em conta estes detalhes, pode-se considerá-lo mais preciso em seus resultados.

5.4.4 Análise das Previsões do Modelo de Simulação

A partir dos resultados do modelo de simulação apresentados na Tabela 6, pode-se afirmar que:

- O maior tempo de execução do ARPS ocorreu na configuração número 1, com 4 h. 48 min., e o menor, na configuração número 12, com 38 min.;
- Todos os 3 fatores estudados, quando aumentados isoladamente, sempre tiveram o efeito de reduzir o tempo de execução do ARPS;
- Das 12 simulações executadas, somente as de números 7 a 12 resultaram em um tempo de execução do ARPS dentro da meta de no máximo 1 h. 30 min. para previsões do tempo reais;
- Nenhuma das configurações com 16 nós atingiu a meta de tempo de execução do ARPS;
- Se a escolha da configuração do *cluster* estivesse limitada às opções da Tabela 6, as que cumpririam as exigências de tempo de execução do ARPS com menor necessidade de recursos seriam: a de número 7 (36 nós, 2 GHz, 100 Mbps) e a de número 9 (64 nós, 1 GHz, 100 Mbps).

Os dados da Tabela 6 estão representados no gráfico da Figura 22 para uma comparação visual dos resultados.

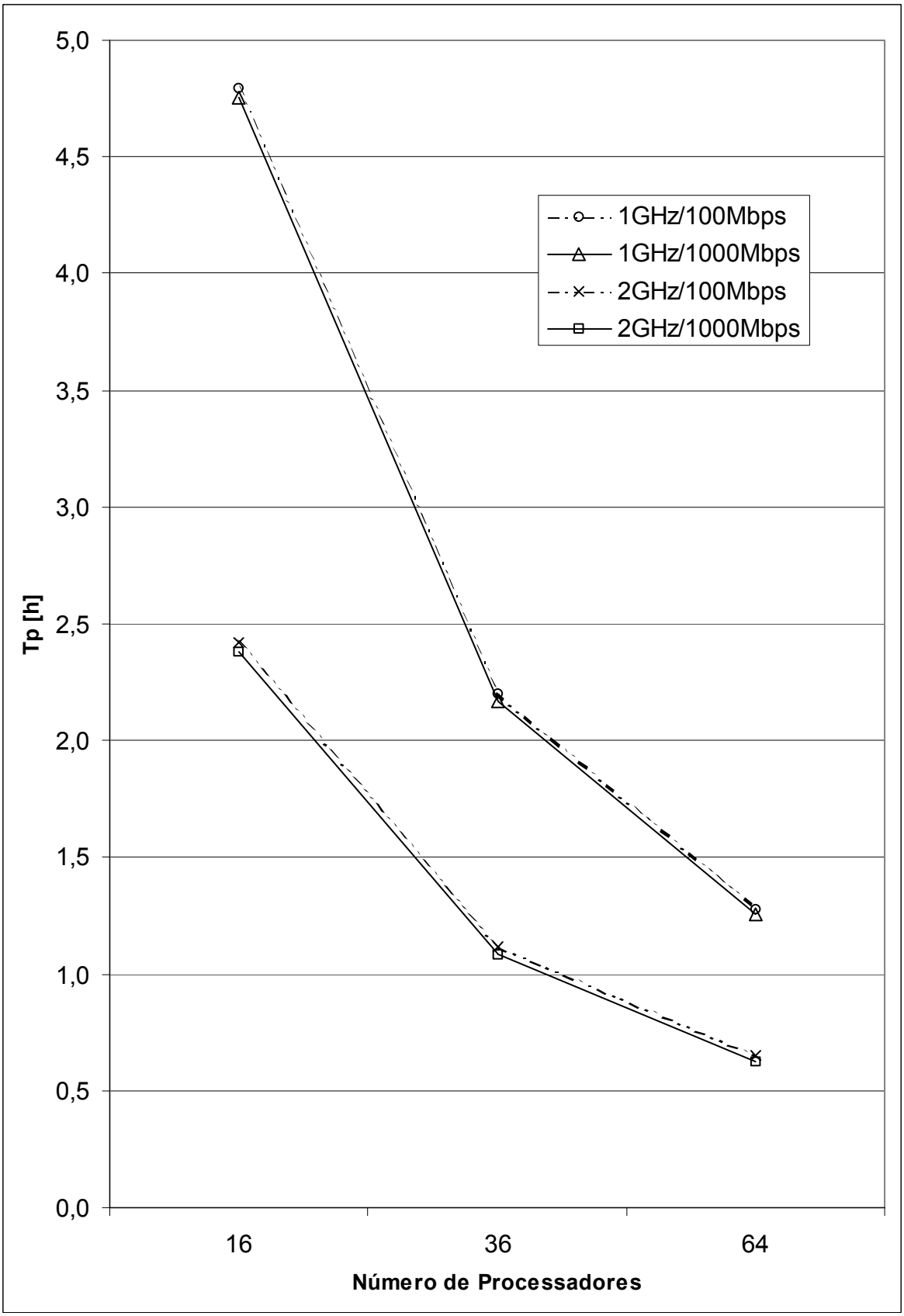


Figura 22 – Gráfico dos resultados do modelo de simulação

A importância relativa destes 3 fatores na variação observada nos resultados da simulação foi avaliada com o método de cálculo de efeitos em projetos de experimentos, apresentado em JAIN (1991). Para concentrar a análise na região do gráfico da Figura 22 em que é cumprida a meta de tempo de execução do ARPS, foram descartadas as simulações com Número de Nós igual a 16, sendo portanto considerados somente as configurações de números 5 a 12.

Foram computados os valores percentuais das variações no resultado da simulação que são explicadas pelo efeito de cada um dos fatores isoladamente (A, B, C), pela interação entre dois fatores (AB, AC, BC) e pela interação entre todos os fatores juntos (ABC). Estes resultados são apresentados na Tabela 10.

Fator(es)	Variação
Vazão da Rede (A)	0,05 %
Clock da CPU (B)	57,89 %
Número de Nós (C)	37,90 %
AB	0,00 %
AC	0,00 %
BC	4,16 %
ABC	0,00 %

Tabela 10 – Fatores simulados e variações explicadas por seus efeitos

Analisando a Tabela 10, pode-se observar que:

- Os dois fatores cujos efeitos isoladamente mais contribuíram para a variação no tempo de execução do ARPS foram: o *Clock* da CPU, com 57,89 %; e o Número de Nós, com 37,90%;

- O fator cujo efeito isoladamente menos contribuiu para a variação no tempo de execução do ARPS foi a Vazão da Rede, com 0,05 %;

- Os resultados tiveram pouca dependência da interação de fatores, sendo que a mais significativa ocorreu entre o *Clock* da CPU e Número de Nós, respondendo por 4,16 % da variação no tempo de execução do ARPS.

6. Conclusões

6.1 Considerações Gerais

Neste trabalho foi estudado o modelo de previsão do tempo *Advanced Regional Prediction System* – ARPS, quanto aos aspectos que afetam seu desempenho quando executado em um *cluster* de computadores.

A partir da literatura existente, foi apresentado um modelo analítico de desempenho do *cluster* para o caso de decomposição unidimensional do domínio. As equações para o caso de decomposição bidimensional foram deduzidas, permitindo o estudo mais preciso de casos reais de aplicação.

Foi desenvolvido um modelo de simulação utilizando a ferramenta Arena, levando em conta alguns detalhes desconsiderados no modelo analítico.

O ARPS foi executado no *cluster* de 6 máquinas existente no laboratório do NURCAD e foram levantadas medidas do tempo despendido para sua de execução em 6 experimentos. Os modelos analítico e de simulação foram validados contra as medidas levantadas no NURCAD.

A seguir, os modelos foram utilizados para um estudo reduzido do impacto de 3 fatores no tempo previsto para execução do ARPS, fazendo-se previsões para configurações do *cluster* que poderão vir a ser construídas. A comparação entre os resultados dos dois modelos apontou diferenças que podem ser explicadas por simplificações assumidas no modelo analítico.

Entre as configurações simuladas neste trabalho, as duas que cumpririam o objetivo de executar o ARPS em até 1 h. 30 min. e exigiriam menor volume de recursos computacionais são: *cluster* com 36 nós de processamento, CPUs com *clock* de 2 GHz e rede Ethernet 100 Mbps; e *cluster* com 64 nós de processamento, CPUs com *clock* de 1 GHz e rede Ethernet 100 Mbps.

O fator cujo efeito isoladamente mais contribuiu para a variação no tempo de execução do ARPS foi o *Clock* da CPU, com 57,89 %. A análise do impacto dos fatores na variação do tempo de execução do ARPS sugere que, para um estudo mais detalhado da região de interesse, bastaria um projeto de experimentos que considerasse apenas os fatores *Clock* da CPU e Número de Nós, mas com um número maior de níveis para cada fator.

Quanto às tecnologias para redes de interconexão disponíveis, observa-se que é grande a disponibilidade de opções no mercado, com grande variação de custo. Algumas ainda são tecnologias proprietárias, porém oferecem alto desempenho, como SCI e Myrinet. Outras são tecnologias que seguem padrões abertos, e estão se tornando alternativas atraentes ao conciliarem desempenho e custo razoáveis, como Gigabit Ethernet.

Os resultados obtidos permitiram concluir que o *cluster* de PCs com sistema operacional Linux ("Beowulf") é uma plataforma que cumpre os requisitos para executar previsões do tempo de qualidade para o Estado de Santa Catarina utilizando o modelo de previsão do tempo ARPS.

6.2 Sugestões para Trabalhos Futuros

Existem importantes linhas de continuação deste trabalho que seriam possíveis, entre as quais destacam-se:

- Executar um maior número de experimentos com o *cluster* do NURCAD, com diferentes tamanhos de grade, para melhor validar os modelos de desempenho;
- Deduzir expressões analíticas para o desempenho do *cluster* levando em conta o uso do comutador;
- Incluir nos modelos de desempenho o *overhead* da pilha de protocolos TCP/IP, que traz um aumento de carga às CPUs relacionado à utilização da rede;

- Comparar o desempenho do ARPS quando executado em máquinas monoprocessadas e em máquinas multiprocessadas (SMP);

- Realizar estudos de escalabilidade do *cluster*, levantando a curva do *speedup* para diferentes configurações;

- Considerar que podem haver comutadores interligados em cascata, ao invés de um único comutador com portas para todos os nós;

- Investigar os benefícios do uso de rede dual – uma rede para dados e outra para controle;

- Investigar o desbalanceamento da carga (quantidade de processamento diferente nas bordas do domínio, quantidade de processamento diferente relacionada ao estado local da atmosfera em uma parte do domínio, etc.) como uma fonte de ineficiência no desempenho do *cluster*, pois é um fator que aparece em casos reais de aplicação do ARPS (BAILLIE, 1997);

- Investigar os efeitos da não-homogeneidade da configuração de hardware dos nós, como, por exemplo, um *cluster* composto por dois grupos de CPUs com *clocks* diferentes;

- Propor e analisar um algoritmo de balanceamento de carga para *clusters* com configuração heterogênea de hardware dos nós; o algoritmo procuraria dimensionar o tamanho do subdomínio atribuído a cada nó conforme o poder de processamento deste;

- Realizar estudos para apontar a melhor alternativa para os componentes do *cluster* em termos de custo/desempenho. Embora este trabalho tenha levantado indicações do desempenho do *cluster* em algumas configurações, uma análise de custos seria necessária para embasar uma futura decisão sobre aquisição de equipamentos.

7. Referências Bibliográficas

1. ÁVILA, Rafael B.; DE ROSE, César A. F.; NAVAUX, Philippe. O. A., et al. Modelagem e Avaliação de Desempenho de Agregados Conectados por Tecnologia SCI. In: 11th SYMPOSIUM ON COMPUTER ARCHITECTURE AND HIGH PERFORMANCE COMPUTING, 1999, Natal, Brasil. **Anais...** p. 107-112. Disponível em: <http://www.inf.pucrs.br/~derose/paginas/public_ano.html>

2. BAILLIE, C.; MICHALAKES, J.; SKŠALIN, R. Regional Weather Modeling on Parallel Computers. **Parallel Computing**. Dezembro de 1997. Vol. 23, Núm. 14, p. 2135-2142. Disponível em: < http://citeseer.nj.nec.com/cs>

3. BAKER, Mark A.; FOX, Geoffrey C.; YAU, Hon W. **Cluster Computing Review**. Novembro de 1995. Disponível em: <ftp://softlib.rice.edu/pub/CRPC-TRs/reports/CRPC-TR95623.ps.gz>

4. BAKER, Mark; BUYYA, Rajkumar. Cluster Computing at a Glance. **High Performance Cluster Computing:** Architectures and Systems. Prentice Hall PTR, 1999. Vol. 1, Cap. 1, p. 3-47. Disponível em: <http://www.csse.monash.edu.au/~rajkumar/cluster/index.html>

5. BANKS, Jerry; CARSON, John; NELSON, Barry, et al. **Discrete-Event System Simulation**. 3ª ed. NJ, EUA: Prentice Hall, 2000.

6. BODEN, Nanette J.; COHEN, Danny; FELDERMAN, Robert E., et al. Myrinet - A Gigabit-per-Second Local-Area Network. **IEEE Micro**, Fevereiro de 1995. Disponível em: <http://citeseer.nj.nec.com/cs>

7. BOGGS, David R.; MOGUL, Jeffrey C.; KENT, Christopher A. Measured Capacity of an Ethernet: Myths and Reality. **Proceedings of the SIGCOMM '88 Symposium on Communications Architectures and Protocols**. Stanford, California, Agosto de 1988. Disponível em: <http://citeseer.nj.nec.com/cs>

8. BRIGHTWELL, Ron; PLIMPTON, Steve. **Scalability and Performance of a Large Linux Cluster**. 1999. Disponível em: <http://www.cs.sandia.gov/~bright/>

9. CAMPBELL, Duncan K. G. **A Survey of Models of Parallel Computation**. Technical Report YCS-278. Department of Computer Science, University of York, 1997. Disponível em: <http://citeseer.nj.nec.com/cs>

10. CAPS – CENTER FOR ANALYSIS AND PREDICTION OF STORMS. In: ARPS Version 4.0 User's Guide. Cap. 11: **ARPS Performance and Optimization on Single and Multiple Processor Systems**. Setembro de 1995. Disponível em: <http://www.caps.ou.edu/ARPS/ARPS4.guide.html>

11. CAPS – CENTER FOR ANALYSIS AND PREDICTION OF STORMS. **About Center for Analysis and Prediction of Storms (CAPS)**. 2001a. Disponível em: <http://www.caps.ou.edu/about_full.htm>

12. CAPS – CENTER FOR ANALYSIS AND PREDICTION OF STORMS. **Technology Transfer**. 2001b. Disponível em: <http://www.caps.ou.edu/technology.htm>

13. CHASE, Jeff; GALLATIN, Andrew; YOCUM, Ken. End-System Optimizations for High-Speed TCP. **IEEE Communications Magazine**. 2001. Disponível em: <http://citeseer.nj.nec.com/cs>

14. CHRISOCHOIDES, Nikos; DROEGEMEIER, Kelvin; FOX, Geoffrey, et al. A Methodology for Developing High Performance Computing Models: Storm-Scale Weather Prediction. **High Performance Computing, 1993:** Grand Challenges in Computer Simulation. The Society for Computer Simulation, 1993. Disponível em: <http://citeseer.nj.nec.com/cs>

15. CHRISTENSEN, Kenneth J.; MOLLE, Mart; LI, Sifang. Comparison of the Gigabit Ethernet Full-Duplex Repeater, CSMA/CD and 1000/100-Mbps Switched Ethernet. **Proceedings of the IEEE 23rd Conference on Local Computer Networks**. Outubro de 1998. p. 336-344. Disponível em: <http://citeseer.nj.nec.com/cs>

16. COMER, Douglas E. **Internetworking With TCP/IP**: Principles, Protocols, and Architecture. 3ª ed. Prentice Hall, 1995. Vol. 1.

17. COMPAQ COMPUTER CORP.; INTEL CORP.; MICROSOFT CORP.; **Virtual Interface Architecture Specification**. Draft Revision 1.0. Dezembro de 1997. Disponível em: <http://www.viarch.org>

18. DANTAS, Mário A. R. **Efficient Scheduling of Paralled Applications on Workstation Clusters**. 1996. Tese de Doutorado - Univ. Birminghan. Disponível na Biblioteca Universitária da UFSC sob registro CETD/GB/0085.

19. DICKENS, Phillip M.; HEIDELBERGER, Philip; NICOL, David M. Parallelized Direct Execution Simulation of Message-Passing Parallel Programs. **IEEE Transactions on Parallel and Distributed Systems.** Outubro de 1996. Vol. 7, Número 10, p. 1090-1105. Disponível em: <http://citeseer.nj.nec.com/cs>

20. FERREIRA, Aurélio Buarque de Holanda. **Novo Aurélio Século XXI:** O Dicionário da Língua Portuguesa. 3ª ed. Rio de Janeiro: Nova Fronteira, 1999.

21. FOSTER, Ian. **Designing and Building Parallel Programs**. Addison Wesley, 1995. Disponível em: <http://www-unix.mcs.anl.gov/dbpp>

22. FOSTER, Ian; KESSELMAN, Carl. Computational Grids. In: 4th International Meeting on Vector and Parallel Processing. **Vector and Parallel Processing -**

VECPAR 2000. Springer-Verlag, 2000. Disponível em: <http://citeseer.nj.nec.com/cs>

23. FRAZIER, Howard. The 802.3z Gigabit Ethernet Standard. **IEEE Network Interactive**. Maio de 1998. Disponível em: <http://www.comsoc.org/ni/public/1998/may/ni802.html>

24. FREITAS FILHO, Paulo José de. **Introdução à Modelagem e à Simulação de Sistemas Discretos**. INE/UFSC, 1997. Apostila de Aula.

25. GEA – GIGABIT ETHERNET ALLIANCE. **Gigabit Ethernet:** 1000BASE-T. 1997. Disponível em: <http://www.10gea.org/Tech-whitepapers.htm>

26. GEA – GIGABIT ETHERNET ALLIANCE. **Gigabit Ethernet:** Accelerating the Standard for Speed. 1998. Disponível em: <http://www.10gea.org/Tech-whitepapers.htm>

27. HAAS, Reinaldo; DOTTA, Daniel; REINERT, Richard R. Avaliação do Modelo de Previsão de Tempo Regional ARPS em Sistema Computacional Distribuído. **Anais do 18º Simpósio Brasileiro de Redes de Computadores.** Belo Horizonte, 2000.

28. HAWICK, K. A.; JAMES, H. A. **Asynchronous Transfer Mode and Other Network Technologies for Wide-Area and High-Performance Cluster Computing**. Technical Report DHPC-079. Department of Computer Science, The University of Adelaide, Dezembro de 1999a. Disponível em: <http://www.dhpc.adelaide.edu.au/reports/index.html>

29. HAWICK, K. A., GROVE, D. A.; VAUGHAN, F. A. Beowulf – A New Hope for Parallel Computing? **Proceedings of the 6th IDEA Workshop**. Rutherglen, Australia, 1999b.

30. HAWICK, K.A.; GROVE, D.A.; CODDINGTON, P.D., et al. **Commodity Cluster Computing for Computational Chemistry**. Technical Report DHPC-073. Department of Computer Science, The University of Adelaide, Janeiro de 2000. Disponível em: <http://www.dhpc.adelaide.edu.au/reports/index.html>

31. HOE, James C.; HILL, Chris; ADCROFT, Alistair. A Personal Supercomputer for Climate Research. **Proceedings of the ACM/IEEE SC99**. Agosto de 1999. Disponível em: <http://citeseer.nj.nec.com/cs>

32. JAIN, Raj. **The Art of Computer Systems Performance Analysis:** Techniques for Experimental Design, Measurement, Simulation and Modeling. John Wiley & Sons, 1991.

33. JOHNSON, Kenneth W; BAUER, Jeff; RICCARDI, Gregory A., et al. Distributed Processing of a Regional Prediction Model. **Monthly Weather Review**. Novembro de 1994. Vol. 122, p. 2558-2572.

34. KELTON, W. David; SADOWSKI, Randall P.; SADOWSKI, Deborah A. **Simulation with Arena**. WCB McGraw-Hill, 1998.

35. MENASCÉ, Daniel A.; ALMEIDA, Virgílio A. F.; DOWDY, Larry W. **Capacity Planning and Performance Modeling:** from Mainframes to Client-Server Systems. Prentice Hall PTR, 1994.

36. MESSAGE PASSING INTERFACE FORUM. **MPI:** A Message-Passing Interface Standard – Version 1.1. 1995. Disponível em: <http://www.mpi-forum.org>

37. MOLLE, Mart L. **A New Binary Logarithmic Arbitration Method for Ethernet**. Technical Report CSRI-298. 1994. Disponível em: <http://citeseer.nj.nec.com/cs>

38. MUELLER, Frank. Implementing POSIX Threads under UNIX: Description of Work in Progress. **Proceedings of the 2nd Software Engeneering Research Forum**. Florida, 1992. Disponível em: <http://citeseer.nj.nec.com/cs>

39. OLIVEIRA, Fábio A. D. de; BARRETO, Marcos E.; ÁVILA, Rafael B., et al. A Comparative Study on Low-level APIs for Myrinet and SCI-based Clusters. **Proceedings of the 4th World Multiconference On Systemics, Cybernetics And Informatics**. Orlando, Florida, Julho de 2000. Disponível em: <http://www-gppd.inf.ufrgs.br/projects/mcluster/papers/papers.html>

40. RIDGE, Daniel; BECKER, Donald; MERKEY, Phillip, et al. Beowulf: Harnessing the Power of Parallelism in a Pile-of-PCs. **Proceedings of IEEE Aerospace**. 1997. Disponível em: <http://citeseer.nj.nec.com/cs>

41. SATHYE, A.; XUE, M.; BASSET, G., et al. Parallel Weather Modeling with the Advanced Regional Prediction System. **Parallel Computing**. Dezembro de 1997. Vol. 23, Número 14.

42. SILVA, Luís Moura e; BUYYA, Rajkumar. Parallel Programming Models and Paradigms. **High Performance Cluster Computing**: Programming and Applications. Prentice Hall PTR, 1999. Vol. 2, p. 4-27. Disponível em: <http://www.csse.monash.edu.au/~rajkumar/cluster/index.html>

43. SOARES, Luiz F. G.; LEMOS, Guido; COLCHER, Sérgio. **Redes de Computadores:** das LANs, MANs e WANs às Redes ATM. 2ª ed. Rio de Janeiro: Campus, 1995.

44. STERLING, Thomas; BECKER, Donald J.; SAVARESE, Daniel, et al. Beowulf: A Parallel Workstation for Scientific Computation. **Proceedings of the 1995 International Conference on Parallel Processing**. Agosto de 1995. Vol. 1, p. 11-14. Disponível em: <http://citeseer.nj.nec.com/cs>

45. TANENBAUM, Andrew S. **Redes de Computadores**. Campus, 1997.