UNIVERSIDADE FEDERAL DE SANTA CATARINA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Marcio Marcelo Piffer

Um Estudo Experimental de Coescalonamento em um Ambiente de Previsão Meteorológica

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

Prof. Mário Antônio Ribeiro Dantas, PhD.

Orientador

Florianópolis, março de 2009

Catalogação na fonte pela Biblioteca Universitária da
Universidade Federal de Santa Catarina

P627 Piffer, Marcio Marcelo
Um estudo experimental de coescalonamento em um ambiente de previsão meteorológica [dissertação] / Marcio Marcelo Piffer ; orientador, Mário Antônio Ribeiro Dantas. - Florianópolis, SC, 2009.
107 f. : il., tabs., grafs.

Dissertação (mestrado) - Universidade Federal de Santa Catarina, Centro Tecnológico. Programa de Pós-Graduação em Ciência da Computação.

Inclui bibliografia

1. Ciência da computação. 2. Cluster (Sistema de computador). 3. Sistemas operacionais (Computadores). 4. Sistemas distribuídos. 5. Tempo - Previsão. I. Dantas, Mário Antônio Ribeiro. II. Universidade Federal de Santa Catarina. Programa de Pós-Graduação em Ciência da Computação. III. Título.

CDU 681

UM ESTUDO EXPERIMENTAL DE COESCALONAMENTO EM UM AMBIENTE DE PREVISÃO METEOROLÓGICA

Marcio Marcelo Piffer

Esta dissertação foi julgada adequada para a obtenção do título de Mestre em Ciência da Computação na área de concentração Sistemas de Computação e aprovada em sua forma final pelo Programa de Pós-Graduação em Ciência da Computação.

_____

Frank Augusto Siqueira, Ph.D.

Coordenador do Programa de Pós-Graduação em Ciência da Computação

Banca Examinadora:

_____

Prof. Mário Antônio Ribeiro Dantas, Ph.D.

Orientador

_____

Frank Augusto Siqueira, Ph.D.

_____

Luis Fernando Friedrich, Dr.

_____

Carlos Barros Montez, Dr.

O tempo e uma certeza:

Quando amamos transmitimos em pequenos atos e gestos,

E as palavras não importam mais;

Quando precisamos de alguém, sentimos sua presença,

E as palavras não têm mais sentido;

Quando nos sentimos sós e abandonados,

Surge uma palavra ou um gesto e descobrimos que nunca estaremos sós.

E no fim apenas a saudade e uma certeza:

Não importa onde estejam, estarão sempre conosco.

(Desconheço o autor)

Aos meus pais Ari e Ana.

Agradecimentos

Ao meu orientador, Prof. Dr. Mário Antônio Ribeiro Dantas, por me dar a chance de tornar isto possível, e por todos os conselhos, dicas e oportunidades. O mestre dos mestres. A todos os amigos e colegas de UFSC e de trabalho, com os quais aprendi muito e compartilhei um ótimo ambiente de amizade ao longo destes anos; especialmente: Vera Lúcia Sodré Teixeira, Adilton Teixeira (Parú), Beth Ulbrich, Daniel Ulbrich, Rafael Speroni, Lilia Speroni, Günter Heinrich Herweg Filho, Raquel Della Giustina, Fabrício Santos Silva, Alex Kühnen, Patrick Padilha, Maiko "J. Fox" Eskelsen, José Carlos Pereira Coninck. Vocês são fantásticos. A todos os meus amigos, cuja presença (real ou virtual) propiciou um alívio para toda tensão desta etapa de minha vida. Ao pessoal do CIRAM/EPAGRI pelo suporte técnico e pela hospitalidade e cortesia com que me apoiaram neste projeto. À minha família, que sempre me apoiou de todas as formas possíveis. Por último, um agradecimento especial à Deise Cristina, esposa e companheira, que soube compreender e ajudar a vencer os vários momentos difíceis que passei nestes últimos tempos, e ser a verdadeira motivação de todo meu esforço e trabalho.

SUMÁRIO

Lista de figuras

Lista das tabelas

Lista de abreviaturas

NPB – NAS (Numerical Aerodynamic Simulation) *Parallel Benchmark*

CCW – Computing on *Cluster*s of Workstations

HPC – High Performance Computing

HPL – High-Performance Linpack Benchmark

NAS – Numerical Aerodymanic Simulation *Parallel Benchmark*

NUMAs - Non-Uniform Memory Architectures

NUMA (Non-Uniform Memory Access)

MPI – Message Passing Interface

LAM/MPI – Local Area Multicomputer/Message Passing Interface

MVAPICH – MPI over InfiniBand and iWARP

WAN – Wide Area Network

UMA – Unifor m Memory Architectures

SMP – Symmetric Multi-Processors

MPP - Massively *Parallel* Processors

CCWs – Computing on *Cluster*s of Workstations

LAN – Local Area Network

SAN – System Area Network

NASA – Numerical Aerospace Simulation Systems Division

SISD – Single Instruction Single Data

SIMD – Single Instruction Multiple Data

MISD – Multiple Instruction Single Data

MIMD – Multiple Instruction Multiple Data

P2P – Peer-to-peer

PEs – Processor Elements

CFD – Computacional Fluid Dymanics

CG – Conjugate Gradient

FT – Fast Fourier Transform for Laplace equation

MG – Multi-grid method for Poisson equation

EP – Embarrassingly Parallel

IS – Integer Sort

SP – Scalar Pentadiagonal Systems

LU – Lower-upper symmetric Gauss-Seidel

BT – Block Tridiagonal Systems

Mflops – Milhões de operações em ponto flutuante por segundo

Mpos/s – Milhões de operações por segundo

ADI – Alternating Direction Implicit

LAPESD – Laboratório de Pesquisa de Sistemas Distribuídos

ATHA-RSAE – Resource Scheduler ATHA Environment

RMI – Remote Method Invocation

EPAGRI/CIRAM – Empresa de Pesquisa Agropecuária e Extensão Rural de Santa Catarina/Centro de Informações de Recursos Ambientais e de Hidrometeorologia

ANOVA – Análise de variância

COTS – Commodity-off-the-shelf

Resumo

Nesta dissertação apresenta-se uma pesquisa de estudo experimental de co-escalonamento em um ambiente real de previsão meteorológica. O ambiente de *hardware* foi caracterizado por uma arquitetura não convencional de *cluster* computacional e o *software* utilizado foi o *benchmark* NAS *Parallel Benchmark* (NPB), um conjunto de aplicações de computação científica utilizado como referência para avaliação de desempenho em diversos trabalhos da área. A escolha pela configuração de *hardware* foi orientada pelo crescimento na utilização desta como uma solução para alto desempenho com excelente relação custo beneficio. Por outro lado, o *benchmark* NAS representa um ambiente de *software* similar aos pacotes empregados na área de previsão de tempo. O ambiente utilizado para os testes foi o da EPAGRI/CIRAM no qual a proposta de coescalamento se mostrou atrativa, posto que a classe de aplicações de *benchmark* denominada S atingiu um desempenho da ordem de 29,39 a 58,23 mais rápido que a classe W. Com o aumento do número de nodos a classe S apresentou uma queda no tempo de processamento 27 vezes mais lenta do que a classe W. Isto quer dizer que se aumentando o número de nodos provavelmente não haverá diferenciação no tempo de processamento entre as classes analisadas.

Palavras Chave: Coescalonamento, ambientes distribuídos, *clusters* computacionais, aplicações de previsão de tempo.

Abstract

This dissertation presents research of an experimental study of scheduling in a real weather forecast environment. The hardware environment was characterized by a 'non-conventional' computing cluster and the software used was the 'NPB', a set of scientific computing applications used as reference for performance evaluation in a variety of papers in the same area. The hardware configuration choice was directed by the growth of this selection as a solution for high performance with an excellent cost-benefit relationship. On the other hand, the NAS Parallel Benchmark represents a similar software environment to the software packages used in the weather forecasting area. The testing environment used was EPAGRI/CIRAM, where the proposal of co-scheduling proved very attractive, furthermore the benchmark applications class, denominated by S, achieved a performance level of 29.39 to 58.23 faster than the W class. Increasing the number of nodes to the S class presented a decrease in processing time 27 times slower than the W class. This means that increasing the quantity of nodes probably won't have differences in the processing time between the analyzed classes.

Keywords: coscheduling, distributed environment, computing clusters, weather forecasting applications.

CAPÍTULO 1

1. Introdução

Arquiteturas computacionais de alto desempenho têm sido testadas por *benchmarks* baseados no seu desempenho propriamente dito e no fluxo de trabalho, o qual está mais focado na área da ciência. A tecnologia de *benchmarks* tornou-se um pilar da avaliação de arquiteturas computacionais. Como o desempenho e a capacidade dos microprocessadores cresceu rapidamente, as exigências das aplicações cresceram na mesma proporção por um maior desempenho no seu processamento. Desta forma, profissionais das mais diversas áreas são requisitados a melhor utilizar esta tecnologia no seu dia-a-dia. Dentre eles podemos citar: geneticistas que precisam de demonstrações rápidas do sequênciamento genético; engenheiros aeronáuticos que precisam testar seus protótipos nas diversas fases de construção de um avião; profissionais ligados diretamente à área da saúde (como médicos, bioquímicos e farmacêuticos) querendo reduzir o uso de cobaias e aumentar a rapidez na criação de remédios utilizando computadores de alto poder computacional em seus experimentos que simulam gerações de indivíduos; meteorologistas que necessitam de uma maior precisão nas previsões de fenômenos meteorológicos, o que pode levar ao salvamento de vidas humanas; etc.

Como resultados do rápido desenvolvimento destas aplicações e dos computadores, os quais são cada vez mais sofisticados, a criação de um conjunto também adequado de *benchmarks* tornou-se uma ferramenta indispensável, porém uma questão controversa e difícil. Os *benchmarks* utilizados na avalição de *softwares* e *hardwares* são os mais variados possíveis. Eles estão sendo desenvolvidos por diferentes organizações para as mais diferentes abordagens. Vários *benchmarks* têm seu foco voltado a diferentes funcionalidades; por exemplo, HPL (LIMPACK, 2008) que tem certa cautela com operações em ponto flutuante de equações lineares. Já o NAS (*Numerical Aerodymanic Simulation Parallel Benchmark* (muito conhecido também como NPB)

Wei et al, (2005), Faraj e Yuan, (2002), Lu, (2004), se preocupa com o desempenho de outros sistemas como a rede de interconexão (ÅHLANDER, 2006). Para os usuários, torna-se uma tarefa difícil a escolha adequada de um ambiente de *benchmark*. Todavia é muito importante que se apresente um conjunto de pontos de referência para a avaliação de diferentes arquiteturas para diferentes campos de aplicação. Neste trabalho, nossa pesquisa está focada em torno das arquiteturas de agregados computacionais *multi-core*, multiprocessados e suas arquiteturas. Também objetivamos avaliar o desempenho de um ambiente real de *cluster*, fazendo a submissão de tarefas aos nodos do mesmo a fim de analisar seu comportamento.

Existe uma gama muito grande de programas de *benckmark* disponíveis para *download* em uma vasta lista de *sites* na Internet. Em nosso trabalho, um ambiente de *cluster* computacional de alto desempenho é avaliado submetendo-se um fluxo de trabalho para que possamos verificar as premissas que acima destacamos utilizando máquinas multiprocessadas. O ambiente utilizado para nossa abordagem foi o ambiente utilizado na EPAGRI/CIRAM (Centro de Informações de Recursos Ambientais e de Hidrometeorologia), que consiste em um agregado computacional do tipo COTS (*commercial off-the-shelf*). Seu *hardware* é formado por doze (12) nós, dentre os quais utilizamos somente sete (07), uma vez que os demais estão dedicados a tarefas únicas na execução dos modelos meteorológicos da organização. Cada nodo do *cluster* possui dois (02) elementos de processamento (EP), dois gigabytes de memória e são todos interconectados por uma rede Gigabit. O único nodo que se diferencia dos demais é o nodo principal, que na organização é conhecido como nodo ADM. Este nodo possui dois megabytes a mais de memória principal, o que totaliza quatro (04) megabytes. Este ambiente possui a distribuição Mandrake do sistema operacional Linux baseado no *kernel* versão 2.6.8.1-12mdksmp instalado em seu nodo ADM e instalado via NFS nos demais nodos. Além disto, neste caso em específico, sobre o sistema operacional estão instaladas todas as aplicações pertinentes ao ambiente de produção voltado à meteorologia.

1.1. Motivação

Na computação de alto desempenho, os *cluster*s aparecem como uma tendência e em conjunção as aplicações de escalonamento dos processos, algo muito presente. Aliados a uma arquitetura de redes de alta velocidade, juntamente com *softwares* de domínio público, incluindo sistemas operacionais, ferramentas e bibliotecas de troca de mensagens, tornam-se uma alternativa economicamente viável, principalmente quando a arquitetura estiver montada em um agregado de *workstations*. Concomitante a isso, o escalonamento de processos, em nosso caso o escalonamento adaptativo em um sistema de computação oportunística, denominado ATHA (HOSKEN, 2003), (MENDONÇA, 2008), (HOSKEN e DANTAS, 2004), (MENDONÇA e DANTAS, 2008), (MENDONÇA, PIFFER e DANTAS, 2008), visa auxiliar a execução de aplicações com grande necessidade de processamento. O coescalonamento com a alocação não-preemptiva de recursos, reunindo a capacidade ociosa do ambiente de alto desempenho, torna possível a execução de tarefas de maneira distribuída e eficiente entre todas as unidades de processamento.

1.2. Objetivos

Existe uma vasta gama de programas de *benckmark* disponíveis para *download* na Internet. Em nosso trabalho, um ambiente de *cluster* computacional de alto desempenho é avaliado submetendo-se um fluxo de trabalho muito próximo ao que os meteorologistas do CIRAM utilizam em seu dia a dia no trabalho de previsão de tempo, a fim de que possamos verificar as premissas que destacamos a seguir, e para isso utilizamos máquinas multiprocessadas. Essas máquinas multiprocessadas fazem parte do ambiente de alto desempenho do CIRAM, que é utilizado para a execução de modelos numéricos meteorológicos. Todos os nodos participantes deste *cluster* computacional são interconectados por uma rede Gigabit Ethernet. Diferentemente de outras abordagens que possuem ambientes de *cluster*s compostos por processadores sequenciais, nosso trabalho está focado em um ambiente totalmente homogêneo, onde a presença de múltiplos núcleos de processamento em um mesmo nodo pode ser o

diferencial em função da densidade de núcleos. Com uma maior densidade pode-se sobrecarregar o subsistema de comunicação interno de cada nodo. O impacto deste efeito poderá alterar o comportamento do desempenho das aplicações em execução no ambiente e com isso o prejuízo computacional tende a ser mais um problema a ser elucidado.

1.3. Objetivos específicos

Os objetivos específicos da presente dissertação são:

- Compreender as propriedades dos *benchmarks* e sua aplicação em determinados domínios, como por exemplo, a capacidade de processamento de um conjunto de funções matemáticas.

- Compreender as arquiteturas de sistemas computacionais paralelos, como um supercomputador, formado por sistemas de agregados computacionais com arquiteturas de sistemas *multi-core* e sistemas multiprocessados.

- Compreender os sistemas computacionais paralelos e investigar suas diferenças e semelhanças em relação a qualquer outra arquitetura de sistema computacional.

- Encontrar e compreender as diferenças e semelhanças entre um sistema de agregado multiprocessado e um sistema representativo multiprocessado.

- Medir e analisar o desempenho de um programa de *benchmark* em um sistema de agregado computacional muito próximo do que é utilizado para execução dos modelos numéricos na EPAGRI/CIRAM que daqui por diante será somente denominada empresa.

- Acompanhar o *benchmark* em suas medições sobre o sistema de agregado computacional, e fazer as possíveis alterações na configuração dos sistemas de *hardware* e *software* de forma a maximizar o desempenho do sistema como um todo.

- Analisar as medições do *benchmark* e sugerir possíveis combinações de sistemas, melhorias e arquiteturas para sistemas utilizados em ambientes de alto desempenho.

1.4. Trabalhos relacionados

De forma a facilitar nossa pesquisa faremos aqui uma delimitação do escopo de nosso trabalho. Sobre uma investigação abrangente na área da arquitetura de computadores, sistemas operacionais, e redes de computadores, porém centrando como objetivo o em ambientes de *cluster* computacional, fizemos uma coletânea dos trabalhos relacionados e seus resultados. Desta pesquisa sobre a literatura relacionada a esses temas culminaram o trabalho aqui exposto.

Os trabalhos em sistemas computacionais paralelos visando mensurar a disputa por recursos, distribuição de processos, afinidade de memória, eficiência na passagem de mensagens, são intensamente investigados e partem do princípio de que os recursos computacionais de uma maneira geral estão disponíveis. Hoje se tem máquinas distribuídas geograficamente em grades computacionais (*Grid Computing*), que por si só não são suficientes no sentido de garantir o aumento da eficiência na distribuição dos processos (DA SILVA, CARVALHO E HRUSCHKA, 2004) . Sendo assim, existem questões como o atraso (*overhead*) da rede em elementos centralizadores e de balanceamento de carga que são pontos fundamentais. Assim, em nosso trabalho não iremos focar essa parte do problema, mas é citado aqui para que tenhamos um melhor entendimento dos pontos que envolvem as questões de paralelização de processos potencializados com a utilização de processadores *multi-cores*, nodos *multi-cores* e multiprocessadores ou nodos de multiprocessadores (POURREZA e GRAHAM, 2007).

Alguns dos trabalhos já pesquisados objetivaram descobrir o efeito do processador e a afinidade de memória em diferentes tipos de cargas de trabalho (*workloads*), sendo executados em sistemas de acesso não uniforme a memória (*Non-Uniform Memory Architectures* - NUMA) e de único *core* e *multi-core.* Em Alan (2006) foi utilizado um dos *micro-benchmarks* de baixo nível presentes no NPB (*Numerical*

Aerodynamic Simulation Parallel Benchmark) (BAILEY, 1991), e algumas aplicações científicas. Neste trabalho foi considerado somente a comunicação intranodos em nodos com 2, 4, e 16 núcleos. Eles registraram um acréscimo de 8% para 12 % no desempenho quando usaram dois *cores* de um mesmo processador, em comparação com o uso de dois *cores* em processadores diferentes. Além disto, foi observado que executando aplicações científicas de larga escala com diferentes parâmetros para processador e afinidade de memória foram confirmados os resultados obtidos com os micro-*benchmarks*.

Há muitos trabalhos em desenvolvimento com o intuito de aumentar a eficiência na comunicação intranodos dos processos MPI (MPICH2, 2008). Eficiência no sentido de reduzir a latência e aumentar a taxa de transmissão. A versão mais recente do LAM/MPI (LAM/MPI, 2008) utiliza métodos eficientes para a comunicação intranodos. A aplicação Nemesis (BUNTINAS, MERCIER e GROPP, 2006) é um recente trabalho que trata de um subsistema de comunicação MPI desenvolvido com a melhoria da comunicação intranodo. Para aumentar a concorrência e reduzir a sobrecarga de bloqueio/desbloqueio foi utilizado o conceito de filas livres de bloqueio (*lock-free queues*). Desta maneira, foi demonstrado que o mecanismo de comunicação Nemesis proporciona menor latência para mensagens com tamanho diferente de zero e alta taxa de transmissão para mensagens maiores que 256 KB quando comparadas a outras implementações MPI. As bibliotecas MPI, LAM/MPI e MPICH são citadas aqui para fins de contextualização, uma vez que no ambiente de alto desempenho utilizado no CIRAM é utilizada a biblioteca MPI.

No trabalho de Chai, Hartono e Panda (2006) foi desenvolvido outro subsistema de comunicação destinado a melhorar a comunicação intranodos para aplicações MPI. Em sua implementação incorporaram o MVAPICH (FOX, 1995), que é uma implementação de alto desempenho para MPI sobre Infiniband, testando-o em um único núcleo e depois em sistemas NUMA multi-*core*. Esta implementação é semelhante ao Nemesis, e utiliza bloqueio para reduzir o *overhead* da rede. O novo projeto também beneficia a utilização eficaz da memória *cache* e o baixo uso da memória principal. Ainda foram considerados os fatores de afinidade do processador para impedir a migração de processos do sistema operacional para os *cores*.

Para demonstrar o novo subsistema de comunicação desenvolvido por Chai, Hartono e Panda (2006), foi utilizado *micro-benchmarks* em comparação à concepção original do MVAPICH, reduzindo a latência de grandes mensagens (maiores que 16 KB) em até 35% e aumentando a comunicação intranodo para mais de 25 % em um computador NUMA de único núcleo.

Em um nó NUMA multi-*core* a latência encontrada de pequenas mensagens trocadas (menores que 16 KB) entre dois *cores* dentro de um mesmo processador também foi menor que entre dois processadores distintos.

A distribuição dos processos para os núcleos dos processadores é importante num ambiente NUMA se existir um padrão específico de comunicação entre os processos. Em Park (2003), foi investigado o efeito de selecionar uma topologia de uma implementação MPI em tempo de execução em um ambiente de grade computacional. Eles consideraram que o ambiente era uma grade com mensagens sendo passadas em uma WAN. Como resultados, conseguiram uma melhoria global de 200% em tempo de execução de uma aplicação selecionando os nodos de acordo com o padrão de comunicação da aplicação, para diferentes agregados computacionais, com diferentes capacidades de comunicação. Esse estudo é igualmente aplicável, de certa forma, dentro de um agregado computacional, onde a comunicação intranodos é normalmente mais rápida, em ordem de grandeza, do que a comunicação internodos para a maioria da interfaces de rede. Assim, a atribuição de processos a um processador é uma questão pertinente ao *cluster*, especialmente quando nele forem utilizados nodos *multi-core* e/ou nodos com multiprocessadores.

Já os trabalhos de Mendonça (2008), Mendonça e Dantas (2008) e Mendonça, Piffer e Viegas (2008) foram desenvolvidos empregando a abordagem de processamento paralelo e oportunístico de Hosken (2003), que faz o coescalonamento adaptativo dos processos em recursos distribuídos. A proposta emprega múltiplas *threads,* tentando aproveitar de maneira mais eficiente todas as unidades de processamento dos *hosts* escravos e a capacidade ociosa de recursos multi-*cores* e de multiprocessadores disponíveis.

1.5. Organização do texto

Visando fornecer o embasamento necessário para contextualizar o tema, para discutir os diversos aspectos que envolvem o mesmo, a estrutura dessa dissertação tem a seguinte organização:

O Capítulo 2 apresenta conceitos relacionados a arquiteturas de ambientes computacionais, seus conceitos, características tais como classificação, tipos, sistemas de computação paralela, sistemas de comunicação, *softwares* relacionados, utilizados e necessários à implementação de um *benchmark*, e os principais conceitos relevantes sobre o tema.

No capítulo 3 são apresentados os principais conceitos relevantes e que abrangem o tema *benchmark* focando o NAS *Parallel Benchmark* (NPB), pois esta é a aplicação que utilizamos juntamente com a protótipo ATHA. Além disso, para melhor entendimento das configurações de um ambiente de computação paralela e conceitos sobre escalonamento de processos em sistemas distribuídos, abordamos esses assuntos neste capítulo.

No Capítulo 4 está o desenvolvimento de nosso trabalho. Mostramos pesquisas que possuem relação com nossa proposta de estudo abordando alguns detalhes de como as mesmas foram implementadas. Na sequência mostramos como nossa abordagem foi implementada e como se dá seu funcionamento.

No Capítulo 5 apresentamos os resultados obtidos na avaliação de nosso experimento no ambiente de *cluster* computacional, e em seguida demonstramos um estudo estatístico para comprovação de nossa abordagem.

No Capítulo 6 apresentamos nossas considerações finais, nossas conclusões sobre o trabalho e estudo propostos, e finalizamos apontando algumas sugestões para perspectivas futuras de continuidade do trabalho.

CAPÍTULO 2: Conceitos, Arquiteturas e Redes de Alto Desempenho

Neste capítulo apresenta-se uma revisão bibliográfica das configurações de redes de alto desempenho, arquiteturas de computadores paralelos e distribuídos e características relativas ao escalonamento de tarefas nos ambientes paralelos e distribuídos. O objetivo é estabelecer uma orientação comum em termos teóricos que serão utilizados ao longo desta dissertação.

2.1. Terminologia

Os termos/conceitos e acrônimos que comumente são usados no campo das redes de interconexão, dos sistemas paralelos e distribuídos muitas vezes podem ter seus significados compreendidos de maneiras diferentes. Sendo assim, nossa intenção aqui é descrever a terminologia e os acrônimos que serão usados neste trabalho de maneira sucinta e que sejam de fácil compreensão a todos.

2.2. Redes de Alto Desempenho

As redes de alto desempenho são caracterizadas pelo amadurecimento das gerações nas redes de computadores. No início, foram as redes conhecidas como Redes de Alta Velocidade (*High-Speed Network*), que foram concebidas para fornecer um conjunto de facilidades para alguns nodos específicos. As redes de alta capacidade (*High-Capacity Networks*), são da segunda geração, e foram concebidas com o intuito de aumentar a capacidade de conexões, entretanto a maioria destas redes foi concebida para ser utilizada em *backbones* (AHMAD e HALSALL, 1993).

2.3. Arquitetura de Computador Paralelo

As arquiteturas de computadores paralelos estão divididas em dois tipos. Um conhecido como sistema **fortemente acoplado** de memória compartilhada, ou ainda multiprocessador e o outro conhecido como sistema paralelo **fracamente acoplado** de memória distribuída, ou ainda multicomputador (também chamado de *cluster* ou agregado computacional) (DASGUPTA, 1990; MAK and LUNDSTROM, 1990). Os diferentes tipos de computadores paralelos são identificados pela forma como estão conectados aos recursos disponíveis dos outros computadores. Em sistemas de memória compartilhada, todos os processadores têm acesso ao espaço do endereço da memória global. Nestes sistemas o processador se comunica por operações nas estruturas dos dados compartilhados usando variáveis compartilhadas para realizar a sincronização. Os sistemas de memória compartilhada são divididos em Arquiteturas de Memória Uniforme (UMA – *Uniform Memory Architectures*), e também são conhecidos como Multiprocessadores Simétricos (SMP – *Symmetric Multi-Processors*), e Arquiteturas de Memória Não Uniforme (NUMAs – *Non-Uniform Memory Architectures*) (DASGUPTA, 1990; MAK and LUNDSTROM, 1990).

2.3.1. Classificação de Flynn

Ao longo dos anos surgiram vários esquemas de classificação de computadores, nenhum dos quais teve ampla aceitação (WEISSMAN, 1999). De todos os esquemas, provavelmente o mais citado é a taxonomia de Flynn (FLYNN, 1972), o qual será adotado neste trabalho.

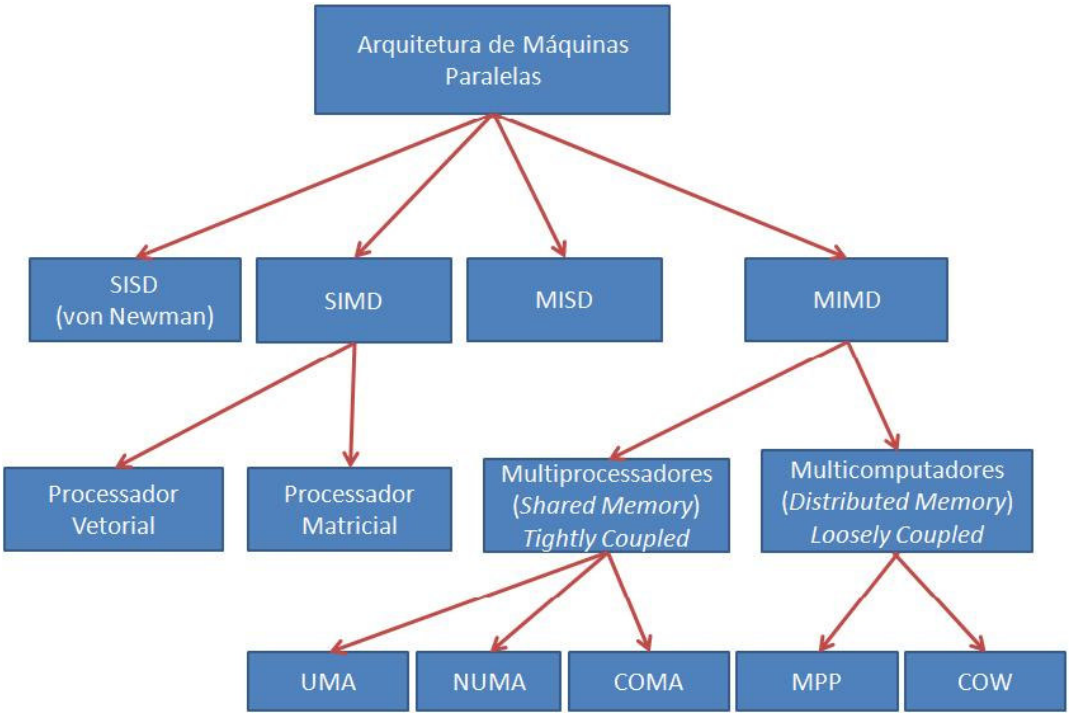


Figura 1: Taxonomia de Flynn (FLYNN, 1972).

Flynn (1972) define quatro classes de computadores, utilizando como critério a quantidade de fluxos de instruções e de dados:

• SISD (fluxo único de instruções e dados, *Single Instruction Single Data*): é a classe que engloba os computadores com um processador;

• SIMD (fluxo único de instruções e vários fluxos de dados, Single *Instruction Multiple Data*): o(s) processador(es) executa(m) a mesma instrução simultaneamente em diferentes itens de dados. Neste caso estão os computadores vetoriais e os matriciais;

• MISD (vários fluxos de instruções e fluxo único de dados, *Multiple Instruction Single Data*): nenhum computador conhecido enquadra-se nesta classe;

• MIMD (vários fluxos de instruções e de dados, *Multiple Instruction Multiple Data*): processadores executam instruções de forma independente um do outro.

Esta classificação, apesar de muito adotada, não é muito específica, uma vez que classifica todos os sistemas distribuídos como MIMD. Em virtude disto, neste trabalho,

será utilizada a proposta apresentada por Flynn (1972), conforme a figura 2, para estender a classe MIMD da taxonomia de Flynn.

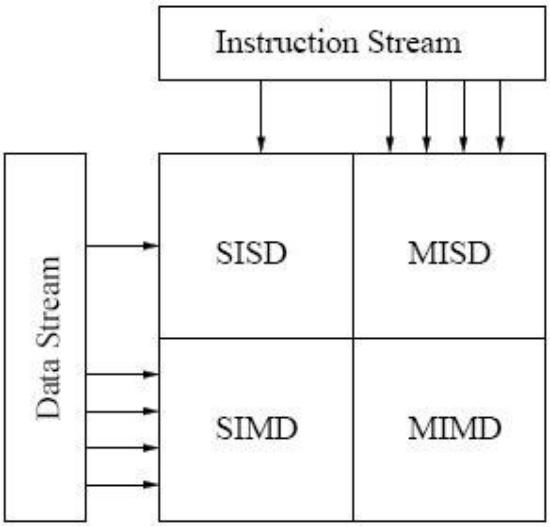


Figura 2: Taxonomia de Flynn.

A figura 2 visa distinguir as arquiteturas com base no fluxo de instruções e no fluxo de dados que são processados simultaneamente.

2.3.2. Sistemas Computacionais Paralelos

Um sistema multiprocessado é um sistema computacional com mais de um processador realizando processamento paralelo. A computação paralela usualmente obtém resultados muito mais rápidos que em um sistema com apenas um processador executando a mesma tarefa. Os processadores em um computador paralelo são hábeis para se comunicarem com os outros processadores que compõem a arquitetura das mais diferentes maneiras. Eles se distinguem em como o processador e a memória irão se comunicar com os outros processadores e memórias dos seus vizinhos. Computadores paralelos são classificados por sua arquitetura de memória, sendo esta memória compartilhada ou distribuída. Um sistema de computação paralelo possui diferentes arquiteturas como, por exemplo, o supercomputador, um sistema de agregado computacional e um sistema de um ambiente multiprocessado (DASGUPTA, 1990).

Todos esses sistemas mencionados podem fornecer um rápido processamento de dados, e todos são para processamento de um conjunto de dados muito grande. Estas arquiteturas podem ainda estar fisicamente em um mesmo ambiente ou podem estar distribuídas geograficamente sem perder seu poder computacional. Devido a isso, são produzidos atualmente sistemas que possuem incorporados à sua arquitetura uma quantidade de núcleos ou processadores muito variada, customizando-se à necessidade da indústria (MAK e LUNDSTROM, 1990).

2.3.3. Redes de Computadores

A idéia principal sobre o que define redes de computadores consiste em termos dois ou mais nodos interligados por uma conexão física. Pode ainda se caracterizar por um grande número de computadores autônomos interconectados por uma ligação física e que provêem compartilhamento de recursos. Outra maneira de ver como se caracteriza uma rede de computadores consiste em termos duas ou mais redes conectadas por um nodo. Em outras palavras, uma rede de computadores pode ser construída pelo aninhamento de redes. Tomando uma coleção de nodos interligados indiretamente pelo aninhamento de redes, será possível para qualquer par de nodos enviarem mensagens para cada sequencia de conexões e nodos interconectados a esta rede (TANENBAUM, 1996). As redes de computadores baseiam-se em alguns princípios os quais incluem organização de protocolos em camadas, comutação de pacotes, roteamento e fluxo de dados (streaming). A diversas técnicas de interconexão de redes possibilitam a integração de redes heterogêneas, e o maior exemplo que podemos citar é a Internet (COULOURIS, DOLLIMORE & KINDBERG, 2005).

2.3.4. Arquiteturas de Memória Compartilhada

Arquiteturas de memória compartilhada, de uma maneira geral, são usadas intensivamente em aplicações comerciais e científicas. A chave destes sistemas é que os

processadores compartilham os recursos disponíveis, como memória e dispositivos de entrada e saída (vide figura 5). Estes projetos permitem maior flexibilidade no compartilhamento e adaptação dos recursos para os mais variados tipos de trabalho (*workloads*). Para computação de alto desemplenho, as aplicações de arquiteturas com memória compartilhada precisam ser ágeis no sentido de apoiar de maneira eficaz um processador em suas tarefas sem tornar-se muito dispendioso. E, para atingir estes objetivos, os fabricantes de grandes sistemas de memória compartilhada possuem duas grandes abordagens: adicionar memória *cache* para esconder a latência da memória e distribuindo a memória para lidar com as demandas cada vez maiores da largura de banda.

2.3.5. Arquitetura de Memória Distribuída

Na arquitetura dos sistemas de memória distribuída, cada processador tem sua própria memória local, a qual não é acessada por qualquer outro processador. Os processos de escrita na memória local não são visíveis para a leitura de outros processadores (vide figura 5). Esta distinção implica profundamente em como os dois tipos de sistemas de memória, distribuída e compartilhada, são programados. A vantagem da memória distribuída é que podemos melhorar a largura de banda desde que as requisições sejam também distribuídas ao longo de vários controladores de memória. Entretanto, a maioria dos sistemas paralelos exige comunicação. Esta comunicação se dá via passagem de mensagem (GRAMA, 2003; DONGARRA, 2003; CULLER and SINGH, 1999). As mensagens são enviadas entre os processadores para sincronização das memórias locais, desta forma permitindo que a escrita seja transparente a todos, ou seja, visível. O controle explícito sobre o fluxo da passagem de mensagens permite também que tenhamos controle sobre o tráfego na rede de interconexão. No entanto, este controle vem com o inconveniente da complexidade da programação, o que reduz a produtividade de maneira geral. Para multiprocessadores de memória compartilhada a comunicação entre diferentes processadores é tratada de forma mais transparente. Um programador pode simplesmente ler e escrever na memória compartilhada. Porém, por via da necessidade de ligação da memória *cache* ao *gap* de memória do processador,

nos deparamos com outro problema. As memórias *cache* locais podem realizar cópias de dados, e se muitos dados estão compartilhados entre os processadores, isto nos leva a um problema de coerência de *cache* (TANENBAUM, 1996; ALAM et al, 2006). Para resolver este problema é aplicado um protocolo de coerência de dados compartilhados, anulando ou atualizando cópias de dados incoerentes. Na próxima requisição de uma cópia incoerente, o dado mais recente precisa se comunicar com a memória *cache* requerente. O uso de protocolos de coerência de *cache* conduz a uma situação em que o fluxo de tráfego na rede de interconexão será gerenciado por padrões de controle de acesso e pela melhora do desempenho das aplicações (HENNESSY & PATTERSON, 2003; CULLER & SINGH, 1999).

2.3.6. Memórias *Cache*

A velocidade das memórias não tem se desenvolvido em passos tão largos com a demanda por largura de banda dos modernos microprocessadores. Memórias com grande capacidade e rápidas são simplesmente muito caras ou muito complexas de se construir. É mais rentável criar a ilusão de uma memória rápida através da inserção de uma pequena, mas rápida memória *cache* (HENNESSY & PATTERSON, 2003; GOODMAN, 1983) entre o processador (CPU) e a memória principal. Uma requisição é enviada de cima para baixo do mais rápido para o mais lento como pode ser visualizado na figura 3 abaixo. Assim, para que a memória *cache* seja eficiente, os programas precisam ser eficientes em manter seus dados críticos na memória *cache*. A isto se chama de princípio da localidade de referência e diz respeito à memória hierárquica, e mais precisamente aos diferentes tempos de acesso aos dados nos diferentes níveis nesta hierarquia (KENNEDY & McKINLEY, 1993).

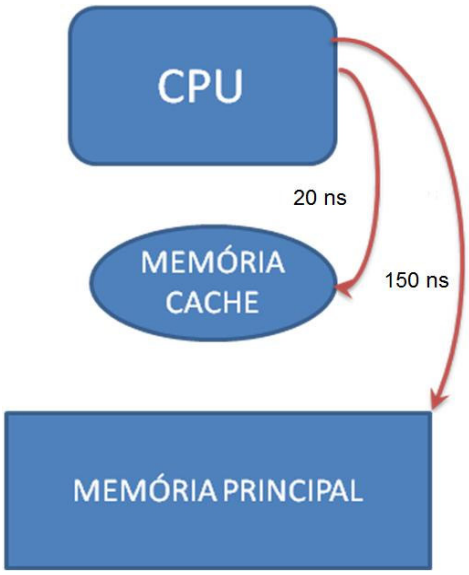


Figura 3: Exemplo ilustrativo dos efeitos da memória *cache*.

Nota-se no exemplo ilustrativo da figura 3 que o tempo de acesso a uma memória *cache* dá-se num tempo de 20 ns e o tempo de acesso à memória principal muito maior na ordem de 150 ns. Isto representa quão mais rápida é a memória *cache* em relação à memória principal.

2.3.7. Multiprocessadores

Uma forma tradicional de melhorar o desempenho no processamento de dados e que ainda está sendo usada é conectar um conjunto de processadores utilizando uma rede de interconexão para formar uma máquina multiprocessada, também conhecida como NUMA (*Non-Uniform Memory Access*) (DANTAS, 2005), (SODAN, 2005), (FOSTER, 1995). A figura 4 na seqüência ilustra o esquema de uma arquitetura NUMA.

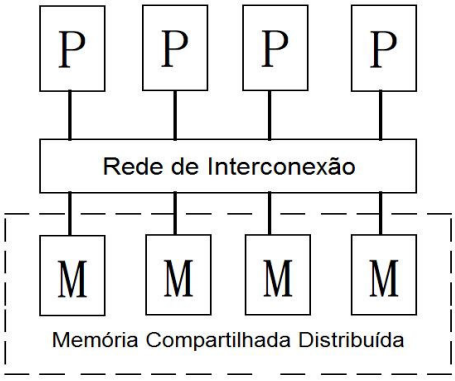


Figura 4: Arquitetura de uma NUMA (*Non-Uniform Memory Access*)

Neste tipo de arquitetura, também conhecida como fortemente acoplada (*tightly coupled*) os processadores cooperam entre si para finalizar as tarefas que a eles são submetidas, o que acontece muito mais rápido do que se tivéssemos apenas um único processador.

2.3.8. Multicomputadores

A arquitetura dos multicomputadores é conhecida como fracamente acoplada (*loosely coupled*). Essas configurações são caracterizadas por possuírem suas próprias memórias locais (vede figura 5). Em outras palavras, o espaço de endereçamento consiste de múltiplos espaços privados de memória, logicamente separados e que, dessa forma, não são diretamente acessíveis por elementos de processamento remotos. Portanto, cada conjunto de elemento de processamento e memória está localizado fisicamente em computadores distintos (SODAN, 2005). A comunicação e sincronização entre os processos em execução são efetuadas apenas por troca de mensagens pela rede de interconexão (DANTAS, 2005). Entretanto, computadores que possuem uma quantidade considerável de memória distribuída necessitam de uma combinação com o espaço de compartilhamento (para balancear o número de processos alocados a cada nodo), e altas cargas de trabalho necessitam de uma combinação com o enfileiramento dos processos para o controle de entrada (SODAN, 2005). Essas

arquiteturas hoje solucionam o problema da deficiência dos multiprocessadores com relação à escalabilidade, porém o atraso (*overhead*) existente na comunicação entre os processos é maior, uma vez que os dados trafegam por uma rede de interconexão (DANTAS, 2005), como em nosso caso.

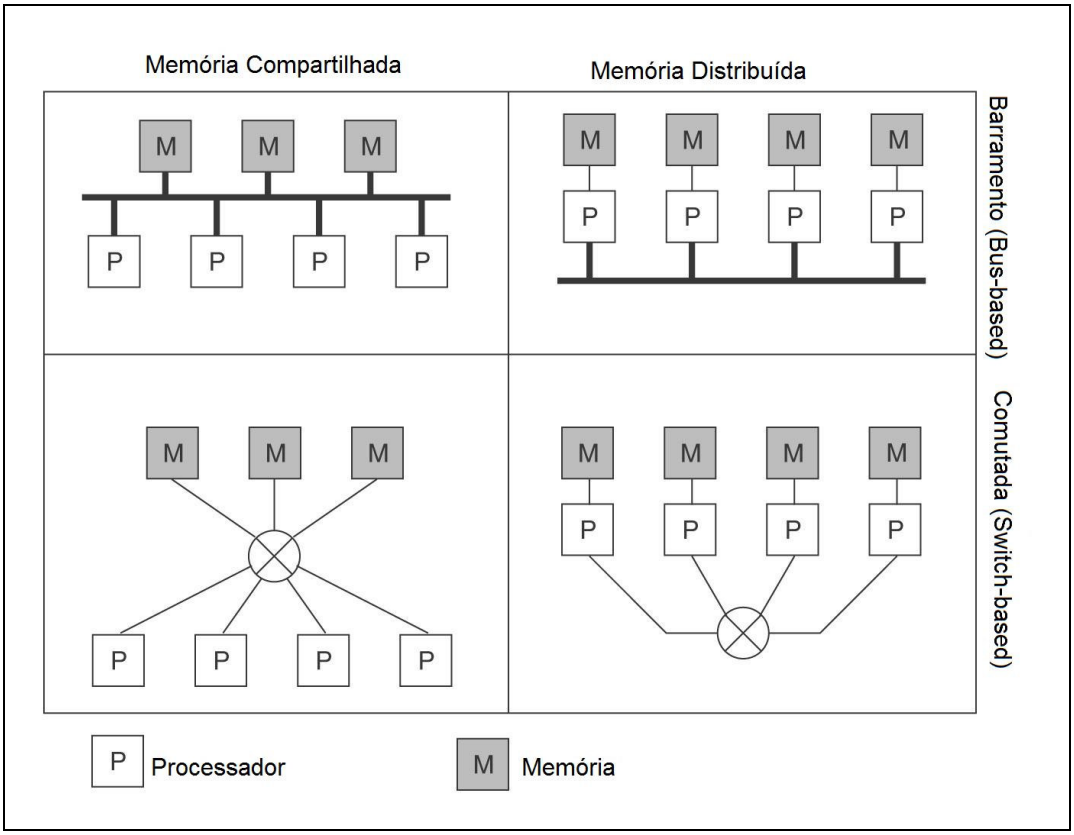


Figura 5: Resumo de multicomputador e multiprocessador

2.3.9. Sistema de Agregado Computacional Paralelo – *Cluster*

Um sistema de agregado computacional paralelo é um sistema geralmente ligado por uma rede de interconexão de um número limitado de dezenas, centenas chegando até a milhares de computadores, estações de trabalho padrão, placas de processador montadas em um *rack*; e eles ainda podem ser monoprocessados ou multiprocessados. Este tipo de sistema tem custo inferior aos computadores convencionais de maior poder computacional, mas ainda podem contribuir com um alto desempenho no

processamento de tarefas, armazenamento de dados, de forma estável e prática (FOX et al, 1995).

Outra forma e uma das mais eficientes maneiras de se realizar grandes tarefas computacionais se dá com a utilização de Super Computadores, também conhecidos como *Massively Parallel Processors* (MPP). Estas máquinas são dotadas de uma coleção de processadores e barramentos internos extremamente rápidos e que utilizam *software* básico desenvolvido especialmente para a arquitetura existente naquela máquina. Assim, obtém-se desempenho próximo ao oferecido pelo *hardware*, ou seja, o *software* é praticamente otimizado ao *hardware* existente.

O *hardware* presente nesses supercomputadores faz com que o preço deste tipo específico de equipamento seja muito elevado. Estas peças são desenvolvidas com tecnologia de ponta. Seu processo de construção é bastante demorado e pode levar mais de dois anos para um projeto ficar pronto (ANDERSON et al, 1995). Assim, se for levada em conta a velocidade com que a tecnologia de *hardware* muda ao chegar ao mercado, a maioria dos componentes de *hardware* de um supercomputador estará disponível para computadores pessoais (PCs), ou já estará defasada. Como exemplo pode-se citar a melhora nos processadores de 32 bits, que a cada 1,5 anos tem seu poder de processamento duplicado (ANDERSON et al, 1995).

Assim, quanto mais rápido se der o término do projeto e o lançamento no mercado, melhor será sua recepção. Mas em contrapartida, as despesas para acelerar o término prematuro do projeto acabam elevando o preço final do produto (ANDERSON et al, 1995). Além do processo de fabricação, outros fatores implicam no custo de um MPP a longo prazo. O *hardware* é bastante inovador e específico, logo o desenvolvimento e a manutenção do *software* básico também elevam o custo de comercialização em larga escala. Assim, com o lançamento de uma nova versão do supercomputador, o seu *software* básico precisa ser praticamente todo refeito.

Em 1991 uma alternativa aos MPPs surgiu: os agregados de computadores pessoais ou *cluster*s de computadores pessoais. Um *cluster* é um conjunto de computadores interconectados por uma rede local ou de sistema. Agregados podem ter desempenho semelhante ao dos MPPs, se o *software* adequado for utilizado.

Através da melhoria nos componentes de *hardware* e *software,* tais como placas de rede, cabos, *switches*, memória, barramento, sistema operacional, e com a melhora nas técnicas de confecção de *software*, foi possível obter *hardware* para CCWs (*Computing on Clusters of Workstations*) que possuísse um desempenho próximo ao do *hardware* de um MPP. A LAN deixou de ser apenas uma interconexão entre computadores e se tornou algo muito mais poderoso, tornando-se o coração do CCW.

Visando especificamente a rede como um item fundamental para CCW, a indústria de *hardware* investiu em tecnologias de rede para proporcionar maior desempenho e confiabilidade. Surgiram então placas de rede extremamente poderosas, dotadas de processadores e memória próprias, voltadas inteiramente a processamento de rede, evoluindo da *Local Area Network* (LAN) para *System Area Network* (SAN). Mesmo com todo o avanço de *hardware,* os CCW não conseguiam ser melhores que os MPP, mesmo nos casos onde a tecnologia utilizada, tanto em processadores quanto em memória, era muito melhor.

A Divisão de Supercomputação Avançada da NASA (NASA *Advanced Supercomputing Division*, também conhecida como, *Numerical Aerospace Simulation Systems Division*) estudou por anos o desempenho de CCW e MPP. Em um de seus estudos sobre o desempenho de computação paralela, conhecido como NAS (*Numerical Aerodynamic Simulation*) (WONG and WIJNGAART, 2003), constataram que as aplicações utilizadas tanto por CCW quanto por MPPs foram um dos pontos principais no ganho de desempenho, além da tecnologia de *hardware* presente nesses computadores.

2.4. Ambientes de Software

No transcorrer desta seção buscaremos apresentar os principais conceitos que dizem respeito aos sistemas distribuídos em relação a ambientes de *software*.

2.4.1. Sistemas Distribuídos

Existem vários tipos de sistemas distribuídos, como *cluster*s, grades computacionais, redes *peer-to-peer* (P2P), sistemas de armazenamento distribuído dentre outros. Esses são alguns dos exemplos, porém existem várias definições na literatura sobre sistemas distribuídos (TANENBAUM, 1996; TANENBAUM, 1990; COULOURIS, DOLLIMORE & KINDBERG, 2005; DANTAS, 2005). Em Coulouris, Dollimore & Kindberg (2005), um sistema distibuído é definido como aquele sistema no qual os componentes localizados em computadores interligados em rede se comunicam e coordenam suas ações apenas passando mensagens. Dentre as diversas definições que estudamos, adotamos a descrita em Dantas (2005) define que os sistemas distribuídos, sob o aspecto de arquitetura de máquinas para a execução de aplicativos, devem ser vistos como configurações com grande poder de escala pela agregação de computadores existentes nas redes convencionais. Além disso, em sistemas distribuídos, a homogeneidade ou heterogeneidade de um conjunto de máquinas, onde cada qual possui sua arquitetura de *software/hardware* executando sua própria cópia de sistema operacional, permite a formação de interessantes SMPs, de MPPs, de *cluster*s e grades computacionais.

Isto vem ao encontro do que estamos utilizando como ambiente de testes para nosso experimento, que consiste em uma aplicação que funciona como um *middleware* gerenciando a submissão de tarefas. Essas tarefas são um conjunto de *threads* do NAS *Parallel Benchmark* (FRUNKIM et al, 2002), as quais são submetidas ao conjunto de computadores do *cluster*. Com isso, podemos observar o desempenho do *cluster* em função da carga de trabalho que lhe é submetida.

2.4.2. A Granularidade dos Dados.

A granularidade da decomposição dos dados possui um impacto significante na eficiência dos programas. Na **granularidade grossa** (*coarse-grained*), a decomposição é feita em um número pequeno de partes grandes. Estes resultados em um pequeno

número de grandes mensagens fazem com que, em determinadas sitiações, ocorra um *overhead* considerável na comunicação. Por outro lado, a decomposição dos dados numa **granularidade fina** (*fine-grained*) faz com que os resultados sejam formados em um número maior de pequenas partes, em muitos casos formando muito mais pedaços do que PEs (*Processor Elements*, em português, Elementos de Processamento). Isto resultada em um grande número de pequenos pedaços, mas facilita enormemente o balanceamento de carga (GRAMA, 2003). Em muitos casos é possível matematicamente derivar em uma granularidade otimizada, onde a decomposição dos dados é feita diretamente pelos programadores, que determinam qual o melhor tamanho das partes. Isto depende, diretamente, do desempenho computacional dos PEs e das características de desempenho da arquitetura de interconexão. Todavia, o programa deverá ser implementado para que a granularidade seja controlada por parâmetros os quais também serão facilmente trocados na compilação ou em tempo de execução. A granularidade é aqui exposta para que tenhamos um melhor entendimento no que tange a distribuição de tarefas a um PE. Desta maneira, salientamos que o tema não fará parte deste trabalho e o intuito aqui é somente de informação.

2.5. Considerações do Capítulo

Este capítulo apresentou uma visão dos conceitos relacionados às configurações de redes de alto desempenho, ambientes de agregados computcionais, arquiteturas de computadores paralelos e distribuídos, suas características, abordando os conceitos de memória compartilhada, memória distribuída e memória *cache*. Focalizamos estes conceitos, pois se fazem necessários à compreensão dos sistemas distribuídos e do ambiente de alto desempenho por nós utilizado deniminado ambiente computacional paralelo (*cluster*).

CAPÍTULO 3: *Benchmarks* e Escalonamento de Processos

Neste capítulo apresentamos e discutimos as principais abordagens encontradas na literatura relacionadas ao escalonamento de processos nos ambientes paralelos e distribuídos e também com o ambiente de *benchmark* NPB. O objetivo é estabelecer uma orientação comum em termos teóricos que serão utilizados ao longo deste trabalho.

3.1. *Benchmarks*

Os sistemas computacionais sempre necessitaram de ajustes em seus parâmetros para que suas funcionalidades sejam exploradas e utilizadas em seu máximo. Não é diferente em outras áreas, mas especificamente no mundo computacional as potencialidades dos *software*s sempre foram exigidas em seu extremo. Desta forma, desenvolveram-se os *benchmarks* para avaliar o desempenho dos sistemas computacionais. Em se falando de sistemas, a NASA (*National Aeronautics and Space Administration*), também conhecido como NASA *Ames Research Center*, desenvolveu um dos *benchmarks* mais conhecidos no mundo científico, chamado de NAS *Parallel Benchmark*, ou simplesmente NPB. Ele foi desenvolvido para avaliar o desempenho dos supercomputadores paralelos. Nosso intuito aqui não é afirmar que o NPB seja o melhor *benchmark* entre tantos. Simplesmente o citamos, pois faz parte do escopo de nosso trabalho. Existem outros *benchmarks* que são utilizados pela comunidade científica no intuito de avaliar o desempenho das aplicações e sistemas, os quais citamos: LINPACK (DANTAS, 1997), Pallas Ping Pong (POURREZA & GRAHAM, 2007), HIMENO (HIMENO, 2008).

3.1.1. NAS *Parallel Benchmark* (NPB)

O NAS *Parallel Benchmark* foi desenvolvido pela NASA *Ames Research Center* para avaliar o desempenho de sistemas paralelos e distribuídos e de

supercomputadores, e este derivou de problemas e aplicações da Fluido Dinâmica Computacional (*Computacional Fluid Dymanics* – CFD) que os cientistas e técnicos da NASA possuiam. Porém, além desta aplicação o NPB foi desenvolvido para avaliar ferramentas e compiladores desenvolvidos na própria NASA e demonstração de paradigmas de programação (MPI, OpenMP, HPF, Java entre outros). O NPB é baseado em seis classes ou tamanhos de problemas: A, B, C, D, W, S, onde A, B, C, e D representam quatro tamanhos de problemas, sendo A o menor e D o maior. O tamanho do problema é o tamanho de um conjunto de dados que será processado pelo *benchmark*. Sendo assim, quanto maior a massa de dados, maior será o tempo de processamento e maior será a requisição por memória. Analisando estas premissas, as classes do *benchmark* se caracterizam da seguinte maneira: a classe A é mais adequada para testes em sistemas com mais de 32 processadores. A classe B se adequa mais a sistemas que possuem entre 32 e 128 processadores. A classe C para sistemas que possuem até 256 processadores. A classe D é para sistemas que possuem mais de 256 processadores. A classe W (*workstation*) é adequada para sistemas com pequena capacidade de memória (abaixo de 32 MB) e a classe S (*sample*) é para sistemas paralelos com menos de 4 processadores. O NPB, até a versão 2.4 possui todo código escrito nas linguagens de programação Fortran77 e C, e por conseguinte, tanto a classe como os processos precisam ser especificados em tempo de compilação. Desde que o Fortran 77 não teve mais alocação dinâmica de memória, somente o *kernel* IS foi implementado na linguagem de programação C e, por conseguinte, tanto as classes como os processos precisam ser especificados em tempo de compilação. Essas linguagens possuem ainda os *kernels* CG (*Conjugate Gradient*), FT (*Fast Fourier Transform for Laplace equation*), MG (*Multi-grid method for Poisson equation*), EP (*Embarrassingly Parallel*), e IS (*Integer Sort*) que são *kernels* paralelos do *benchmark* que são executados em potência de 2, esta sendo o número de processos (1, 2, 4, 8, 16...). No entanto, SP (*Scalar Pentadiagonal Systems*), LU (*Lower-upper symmetric Gauss-Seidel)* e BT (*Block Tridiagonal Systems*) são pseudo-aplicações de *benchmarks* executados em números quadrados de processos (1, 4, 8, 16...). A unidade de medida é Mflops (milhões de operações em ponto flutuante por segundo), mas o IS está em Mpos/s (milhões de operações por segundo) (JIN, FRUMKIN & YAN, 2008; BAILEY, 1991).

Neste trabalho estudamos a implementação 3.0 do NPB (FRUMKIN, 2002). Esta versão destina-se a executar, com pouca ou nenhuma modificação seus processos, a fim de aproximar o desempenho mensurado ao desempenho que um típico usuário necessita para obter um programa paralelo portável.

3.1.2. A Aplicação BT

Dentre os *benchmarks* pesquisados optamos por utilizar o NPB, mais especificamente a aplicação BT (*Block Tridiagonal Systems*), pois esta possui um fluxo de dados (*workload*) que se assemelha muito com o fluxo de dados dos modelos numéricos utilizados pela empresa na produção da previsão de tempo. Dentre as classes que a aplicação BT possui utilizamos somente as classes S e W, sendo classe W, dentre ambas, que possui maior demanda de poder computacional, aproximando-se da demanda que os modelos numéricos utilizados pela empresa requerem.

Na sequência descrevemos de forma sucinta, para darmos uma visão geral, a aplicação BT e alguns detalhes que encontramos e que são relevantes abordarmos.

- BT, sigla de *Block Tridiagonal*, é uma aplicação CFD (Fluidodinâmica Computacional, em inglês *Computacional Fluid Dynamics*) que simula o uso de um algoritmo implícito para resolver equações tridimensionais (3-D) compreessíveis de Navier-Stokes. A solução diferencial finita para o problema é baseada em uma fatoração aproximada de um ADI (*Alternating Direction Implicit* ou método totalmente implícito), que separa as dimensões x, y e z com uma técnica de volumes finitos e provê formas altamente eficientes de se resolver equações diferenciais parciais dependentes de uma variável e em duas ou mais dimensões (FORTUNA, 2000). Em conjunto a técnicas de discretização, resulta um sistema de equações de Bloco Tridiagonal com blocos 5 x 5, ou penta diagonal, e o mesmo é resolvido sequencialmente juntamente com cada dimensão (JIN, FRUMKIN & YAN, 2008; BAILEY, 1991; LINPACK, 2008; FARAJ, 2002; LU, 2004), necessitando, portanto, de muita memória, além de fazerem testes de utilização da memória *cache* e registradores, através de rotinas de laços com tamanho ajustáveis. Essa última parte não foi foco de nossa investigação, mas cabe aqui o

registro de sua existência. Em nosso experimento as matrizes chegaram a tamanhos da ordem 24 x 24 x 24. Num primeiro momento, pensando-se em um *benchmark* que chega a produzir aleatoriamente matrizes de ordem 700 poder-se-ia realizar o execução dos cálculos sem maiores problemas no agregado computacional. Porém, em se tratando em um ambiente computacional dedicado e em produção não seria sensato exaurir seus recursos prejudicando os resultados da execução dos modelos meteorológicos.

3.2. Escalonamento de Processos em Sistemas Paralelos e Distribuídos

Mencionamos nas seções do capítulo anterior, a importância dos sistemas paralelos e distribuídos. Segundo Casavant & Kuhl (1988), existem duas propriedades que devem ser consideradas na avaliação de qualquer sistema de escalonamento. A primeira diz respeito à satisfação dos consumidores em relação ao quão bem o escalonador gerencia os recursos envolvidos (*performance*). A segunda diz respeito à satisfação dos consumidores em termos de quão difícil ou oneroso é o acesso a gestão dos recursos (*eficiência*).

3.2.1. Escalonamento de processos (*Job Scheduling*)

Escalonamento de processos (*Job Scheduling*) é um conceito que se confunde com o próprio conceito de escalonamento (*scheduling*) e significa determinar quando e onde os processos (*jobs*) serão executados e quantos recursos serão alocados para a execução destes processos. Além disso, todos os processos devem ser executados de forma organizada e contínua e os lotes de processos devem seguir uma sequência eficiente para alcançar o desempenho esperado. Isto acontece tipicamente uma camada acima do sistema operacional em que o desempenho do escalonamento dos elementos de processamento é permitido para a execução dos processos (SODAN, 2005).

De maneira geral, o escalonamento de processos tem sido descrito de diferentes maneiras na literatura (DANTAS, 2005), (CASAVANT & KUHL, 1988). No que diz respeito a processos distribuídos, (CASAVANT & KUHL, 1988) reitera que, numa ampla visão das funções de escalonamento, os processos são vistos como recursos que gerenciam recursos. Diz ainda que o gerenciamento destes recursos é basicamente um mecanismo ou política usada para gerenciar o acesso dos consumidores aos recursos. E mais, um sistema de escalonamento possui três componentes principais, como descritos na figura 6. Os recursos são os elementos de processamento (EPs), memórias, rede de comunicação. Os consumidores dizem respeito às aplicações e processos submetidos para execução ao ambiente computacional. Finalizando, há o escalonador de processos com seus mecanismos e políticas utilizadas na distribuição dos mesmos aos seus respectivos processadores.

Figura 6: Sistema de escalonamento segundo Casavant e Kuhl (1988).

Casavant e Kuhl (1988) dizem ainda que quando um sistema de escalonamento é avaliado, duas propriedades devem ser levadas em consideração: 1) a satisfação dos consumidores com relação ao modo como o escalonador gerencia os recursos disponíveis (desempenho), e 2) a satisfação dos consumidores em termos de quão difícil ou custoso é acessar o mecanismo de gerenciamento de recursos (eficiência). Em outras palavras, os consumidores desejam estar aptos a acessar de forma rápida e eficiente o recurso atual em questão, mas não desejam sofrer o atraso de utilizar a função de escalonamento.

Casavant e Kuhl (1988) propuseram ainda uma taxonomia de escalonamento que pode ser observada na figura 7.

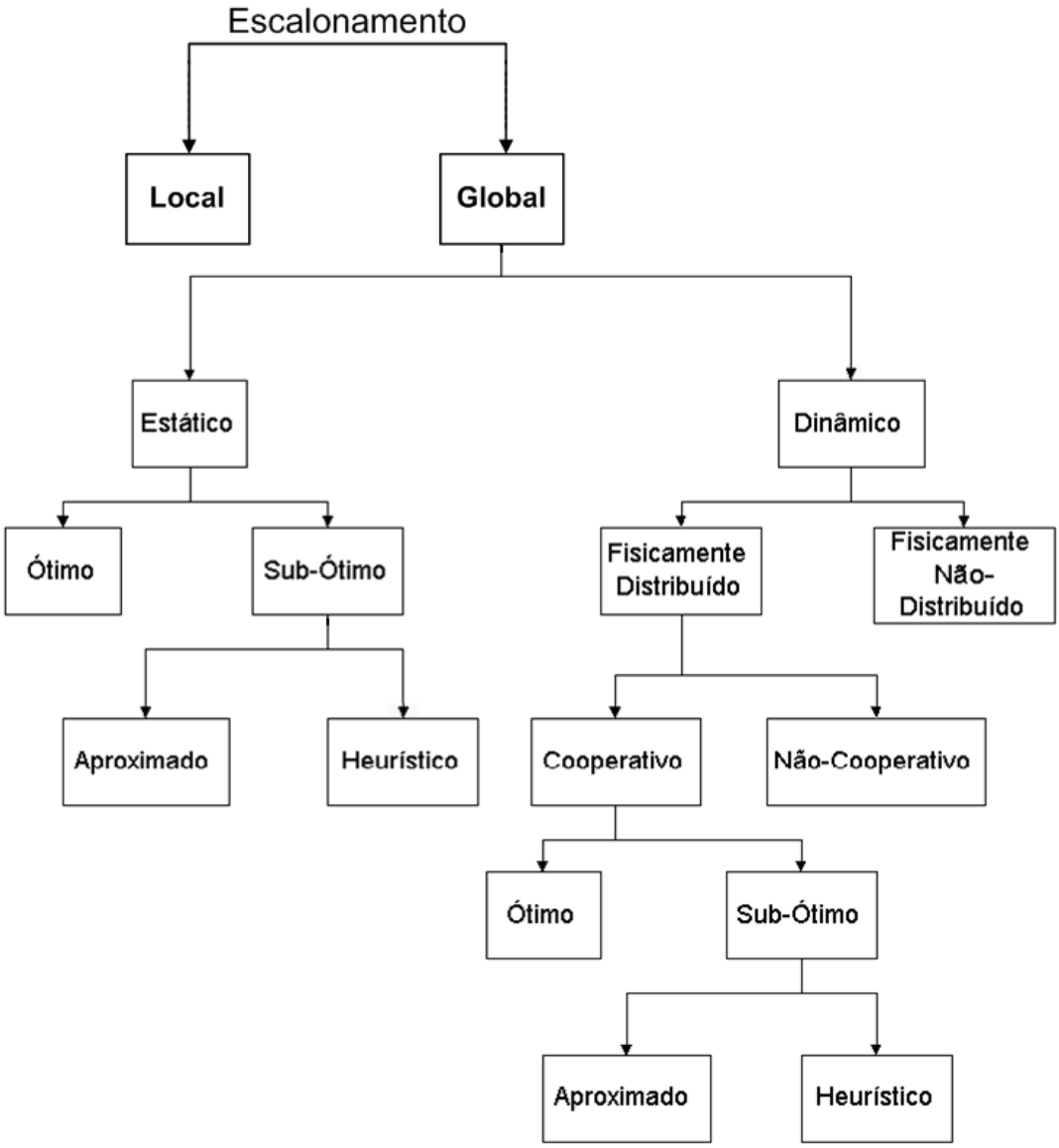


Figura 7: Taxonomia de escalonamento de processos proposta por Casavant e Kuhl (1988) adaptada de Pinto (2004).

Conforme a figura 7, Casavant e Kuhl (1988) definiram os possíveis tipos de escalonamento iniciando com os métodos de escalonamento local e global. O escalonamento local diz respeito às fatias de tempo (*time slices*) atribuídas de um processador aos processos. De outra maneira, o escalonamento global refere-se ao problema de decisão em onde os processos serão executados, sendo desta forma, seus

métodos aplicáveis aos sistemas distribuídos (DANTAS, 2005). Deste ponto em diante nosso foco estará voltado à subdivisão dos dois grupos a partir do método de escalonamento global.

3.2.2. Escalonamento Estático versus Dinâmico

O subnível logo abaixo do escalonamento global é a escolha entre o escalonamento estático e o dinâmico (DANTAS, 2005), (CASAVANT & KUHL, 1988). Neste nível a escolha indica o momento em que o escalonamento da aplicação será feito e como os processos serão distribuídos. O escalonamento estático diz respeito à decisão de como os processos serão distribuídos entre os processadores, ou seja, **como será feito o balanceamento das cargas antes da execução**. Esse método é empregado em aplicações das quais já possuímos informações acerca dos tempos de execução e informações acerca dos recursos computacionais disponíveis, como elementos de processamento, em tempo de compilação. Além disso, esta abordagem é mais eficiente em configurações de *hardware* homogêneo. Nossa abordagem pode ser classificada como sendo do tipo estática, uma vez que o número de nodos que participaram do processamento da aplicação é pré definido no início da execução. O escalonamento estático, por sua vez, divide-se em ótimo e subótimo, porém, não abordaremos aqui este ramo da classificação de Casavant e Kuhl (1988).

3.2.3. Escalonamento Distribuído *versus* Não Distribuído

A partir do escalonamento dinâmico definido por Casavant e Kuhl (1988), uma nova divisão é feita definindo o escalonamento como distribuído e não distribuído. Isto diz respeito à **distribuição ou não da responsabilidade do escalonamento**. Como em nosso caso, um único nodo pode ser responsável pela tarefa de escalonar os processos entre os diversos nodos de um sistema distribuído, sendo que nesse caso o escalonamento é não distribuído. A tarefa de escalonamento, ainda pode ser executada

por mais de um nodo, e esses vários nodos podem também cooperar entre si. Em nosso caso o escalonamento é não distribuído, pois a tarefa de escalonamento fica a cargo no nodo *master* do *cluster* computacional por nós utilizado.

3.2.4. Escalonamento Adaptativo *versus* Não Adaptativo

Casavant e Kuhl (1988) definem na sequência de sua taxonomia o escalonamento como adaptativo ou não adaptativo. O escalonamento se dá de maneira adaptativa quando o algoritmo **modifica dinamicamente suas regras** de acordo com o comportamento atual ou anterior do sistema de escalonamento, ou seja, o sistema tem a **capacidade de se adaptar ao meio**. De outro lado está o escalonamento não adaptativo, o qual não modifica o controle básico de seu funcionamento com base no histórico das atividades do sistema. Ou seja, em função do estado em que o sistema se encontra ele **pode ou não adaptar seu funcionamento** básico em função do histórico das atividades que o sistema possui.

Em nosso caso, podemos classificar nosso trabalho como **não adaptativo**, pois nosso ambiente possui uma arquitetura específica e nossa aplicação não está configurada para se adaptar à entrada de novos nodos, o que deverá ser realizado manualmente.

3.2.5. Escalonamento Preemptivo versus Não Preemptivo

O escalonamento pode ser preemptivo ou não preemptivo de acordo com sua capacidade de **permitir ou não** que alguma tarefa possa ser transferida de um nodo para outro em tempo de execução. Isto é uma tarefa um tanto árdua, pois o sistema precisa guardar o estado dos processos e transferi-los em conjunção ao seu estado momentâneo. Já no escalonamento não preemptivo não é permitida a migração de processos, uma vez que as tarefas não devem ter sido iniciadas em outros nodos do sistema.

Nossa abordagem classifica-se como **não preemptiva**, pois as tarefas não podem ser transferidas entre os nodos após terem sido iniciadas em nodos *workers*.

3.3. Considerações finais

Neste capítulo abordamos a tecnologia de *benchmarks*, mais especificamente o NAS *Parallel Benchmark,* focando a aplicação BT por esta ser aplicação utilizada em nosso trabalho juntamente com duas classes de problemas da mesma . Em seguida foram apresentadas características relativas ao escalonamento de processos em ambientes paralelos e distribuídos, e a abordagem de escalonamento de processos proposta por Casavant e Kuhl (1988) e sua taxonomia.

CAPÍTULO 4: Proposta de Estudo Experimental

Este capítulo apresenta a proposta de nosso trabalho com uma descrição do modelo proposto e sua relação com os trabalhos descritos anteriormente com a intuito de especificar as principais características de funcionamento do modelo proposto.

4.1. Introdução

A proposta de estudo experimental apresentada neste capítulo representa uma contribuição diferencial de coescalonamento de processos, baseada na necessidade de uma aplicação de previsão de tempo. A idéia é estudar a distribuição de processos em agregados de computadores, visando indicar uma tendência de desempenho para as aplicações utilizadas na empresa baseadas na distribução dos processos, neste caso os modelos numéricos de previsão de tempo. Dentre as diversas abordagens de *benchmarks* existentes decidimos por utilizar o NAS *Parallel Benchmark* em função deste ser uma das aplicações mais utilizadas e consolidadas para a avaliação de desempenho de ambientes e aplicações computacionais. Além disso, sua distribuição e a carga de trabalho que este *benchmark* utiliza para esta avaliação, especificamente a aplicação BT, é muito próxima da carga utilizada na empresa. Em trabalhos anteriores do nosso grupo de pesquisa (LAPESD, 2008) foi desenvolvido um protótipo denomindado ATHA (HOSKEN, 2003; HOSKEN & DANTAS, 2004) e ATHA-RSAE (MENDONÇA, PIFFER & DANTAS, 2008), cuja finalidade foi avaliar e reduzir ao mínimo o custo de processamento. Sendo assim, decidimos por combinar as duas tecnologias e termos uma melhor visualização dos resultados desta aliança.

Devido às limitações de uso do ambiente computacional e das limitações no uso do espaço da empresa utilizamos apenas as classes S e W em nosso experimento e os resultados desta experiência estão nesta deissertação no capítulo 5.

4.2. A abordagem proposta

O projeto da abordagem proposta teve como linha base de raciocínio os trabalhos propostos em (MENDONÇA, 2008), (HOSKEN & DANTAS, 2004), (MENDONÇA & DANTAS, 2008) e (MENDONÇA, PIFFER & DANTAS, 2008) os quais foram desenvolvidos pelo nosso grupo de pesquisa do Laboratório de Pesquisa de Sistemas Distribuídos (LAPESD, 2008). A abordagem é do tipo Cliente/Servidor com um nodo mestre (*master*) submetendo tarefas a um nodo escravo (*worker*). As abordagens acima citadas atribuem maior ênfase à heterogeneidade do agregado computacional aproveitando os ciclos ociosos dos nodos presentes em uma rede. Em nosso caso, o ambiente é totalmente homogêneo, onde a arquitetura computacional é a mesma para todos os elementos de processamento interconectados por uma rede Gigabit. A aplicação foi testada com a implementação da aplicação BT (*Block Tridiagonal Systems*) do NAS *Parallel Benchmark* (ou NPB) mais duas classes de algoritmos: S e W. Esse *benchmark* foi testado em sua versão traduzida da linguagem de programação C para a linguagem de programação Java, sendo esta a versão 3.0 do *benchmark* NPB, onde todos os algoritmos são *threads* (FRUMKIN *at al.*, 2002).

4.3. Aspectos Relativos à Implementação

Em função de a aplicação BT ser implementada em Java utilizando *threads* e as abordagens ATHA (HOSKEN, 2003) e ATHA – RSAE (MENDONÇA, 2008) utilizarem a mesma linguagem de programação, neste caso a linguagem de programação Java, fizemos uma primeira tentativa no intuito de utilizar essas abordagens para comprovar nosso experimento. Infelizmente não houve sucesso, pois se tornou uma tarefa onerosa e de difícil entendimento a interpretação do módulo *master* destas aplicações. Então, decidiu-se adotar RMI por ser nativo da linguagem Java e para poder interagir com o objeto localizado na máquina remota como se este objeto fosse local. Os resultados mostraram que não foi uma boa escolha, pois na submissão dos processos o ATHA enviava um objeto para processamento no módulo escravo. Isso dificultava o

processo de comunicação utilizando RMI e em adição acabava degradando a largura de banda chegando ao ponto de ser inviável sua utilização. Então, a utilização da tecnologia RMI foi descartada.

Uma nova tentativa utilizando *sockets* foi iniciada a qual se mostrou mais viável em termos de facilidade de trabalho com os dados e rapidez no tempo de resposta. Nessa nova abordagem utilizando *sockets* faz-se o envio de uma instância do objeto para o nodo escravo (*slave*) e esse objeto é então executado remotamente. Com a mudança da tecnologia utilizada no envio dos processos ao nodo remoto, também foi necessário modificar à máquina de estados de nossa abordagem inicialmente entendida das abordagens ATHA (HOSKEN, 2003) e ATHA-RSAE (MENDONÇA, 2008). Como nossa intenção é avaliar o desempenho do *cluster*, fazendo a submissão de tarefas aos nodos do *cluster* para podermos avaliá-lo e por estarmos utilizando um ambiente computacional dedicado a execução de modelos meteorológicos, em produção, não poderíamos utilizá-lo à exaustão a ponto de exaurirmos seus recursos comprometendo a execução dos modelos numéricos. Sendo assim, extraímos alguns métodos (*hello, olleh, get, result*) das implementações das abordagens (HOSKEN, 2003), (MENDONÇA, 2008), (HOSKEN; DANTAS, 2004) com o intuito de melhorar o desempenho da aplicação. Esses métodos foram desenvolvidos para trabalhar com computação oportunística, em que os *workers* podiam se desconectar e novos *workers* podiam enviar requisições para conectarem-se ao ambiente de computação oportunística a qualquer instante. Uma vez conectados, esses novos voluntários solicitam tarefas ao módulo *master* a fim de contribuir no processamento da aplicação. Existindo tarefas pendentes na fila de execução, essas serão submetidas a esse novo voluntário. Isso tornava o ambiente bastante mutável, como mostram os trabalhos (HOSKEN, 2003; MENDONÇA, 2008). Com as mudanças que propusemos em relação às versões anteriores da abordagem ATHA (HOSKEN, 2003), nossa abordagem pode ser observada na figura 8.

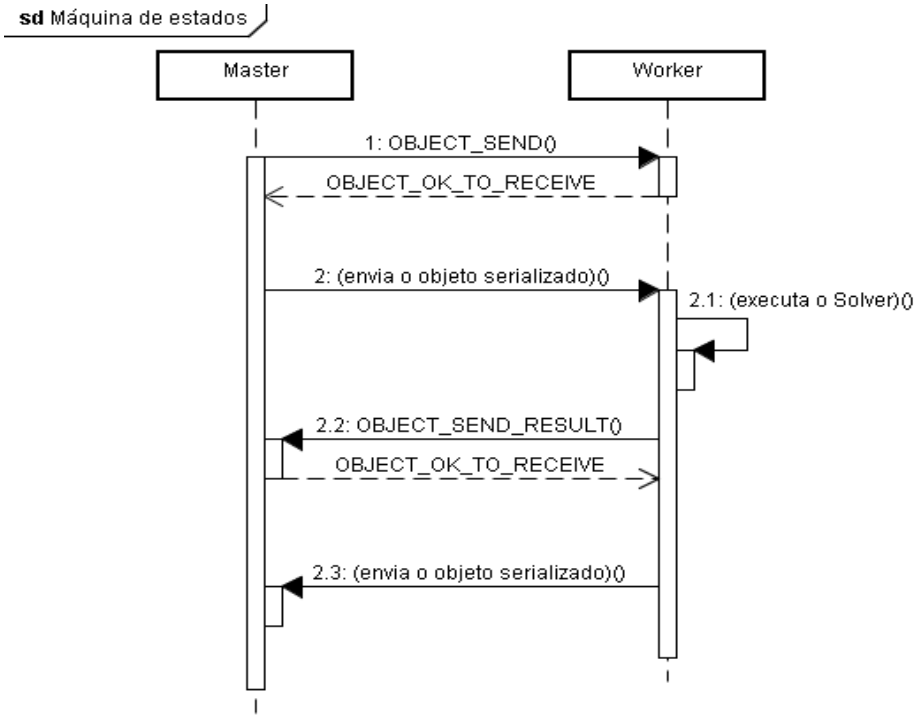


Figura 8: Diagrama de sequência da Máquina de Estados de nossa abordagem.

As mudanças realizadas na máquina de estados, conforme a figura 8, se fizeram necessárias porque a abordagem proposta já não era mais um ambiente de computação oportunística. Mais ainda, a implementação do ATHA foi feito com a idéia de computação oportunística, porém se mostrava pouco flexível para a inserção de novos problemas, como o NPB em nosso caso. Decidiu-se então por adotar a idéia principal do ATHA que era a execução remota e o escalonando de processos, não utilizando o conceito de computação oportunística, mas com a migração de processos de um nodo para outro, porém de maneira **não preemptiva**.

Assim sendo, se fazia necessário o escalonamento dos processos. Como a aplicação BT do NPB é composta de várias classes (. class do Java), a necessidade de serializar o objeto enviado ao módulo *worker* tornou-se prioritária, pois o mesmo era muito grande. Em outras palavras, nossa aplicação toma o objeto que é serializado, o envia ao módulo escravo, o qual é executado remotamente. Tendo em vista que o ambiente é totalmente homogêneo e controlado, então buscamos uma alternativa que nos possibilitasse a distribuição dos processos aos nodos do *cluster*.

O diagrama de sequência da máquina de estados, que pode ser visto na figura 8, mostra-nos o modelo proposto para nosso experimento. Por ser não preemptiva, a aplicação inicia-se com o módulo mestre enviando uma mensagem OB JECT_SEND aos nodos escravos, pedindo autorização para enviar um objeto. Ao receberem esta mensagem, os nodos escravos enviam uma mensagem de resposta, OBJECT_OK_TO_RECEIVE para que o nodo mestre envie o objeto. Na sequência, o nodo mestre inicia a serialização do objeto e sabe que a única mensagem que ele pode receber e aceitar do *worker* (escravo) é OBJECT_SEND_RESULT. Esse método pede permissão para enviar o objeto de volta, e o nodo *master,* através da mensagem OBJECT_OK_TO_RECEIVE, envia a confirmação de que aceita o recebimento. Então para finalizar, o nodo escravo serializa novamente o objeto processado e o envia ao nodo mestre.

Se o *worker* rejeitar, por qualquer motivo, a requisição de envio de tarefas OBJECT_SEND, aquele *worker* não fará parte do agregado de processamento daquela tarefa e seu poder de processamento não será usado, uma vez que o *master* não reenviará a requisição num próximo momento. Em outras palavras, não há uma nova tentativa de reenvio de tarefas e esse *worker* perdido.

Podemos notar aqui que uma parte da abordagem ATHA (HOSKEN, 2003), e ATHA-RSAE (MENDONÇA, 2008) foi suprimida. Justamente a parte responsável pelo gerenciamento das tarefas. Assim, a execução das *threads* enviadas pelo NPB aos nodos do *cluster* são executadas em paralelo em diferentes *workers*.

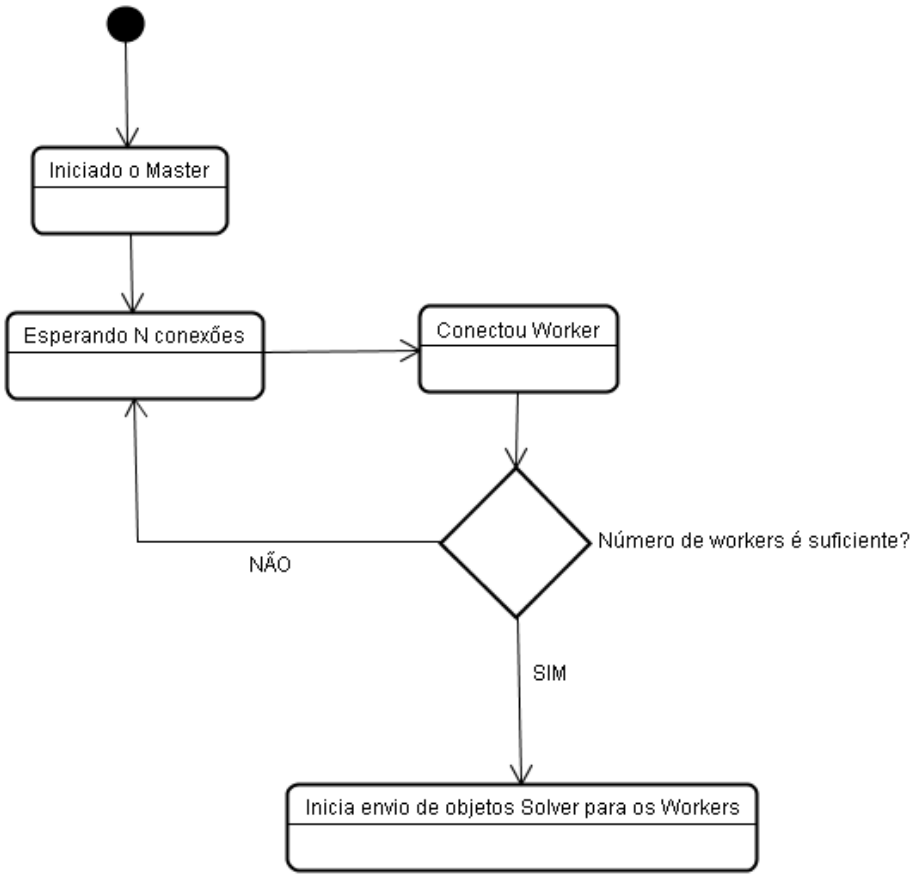


Figura 9: Fluxo de dados de cada nodo escravo (*worker*)

Após o início da operação do *master*, o mesmo espera por conexões dos *workers*. Estas conexões são feitas via requisição do *master*, OBJECT_SEND e uma vez conectado o número de *workers* pertencente àquela rede será feita uma avaliação visando saber se o número de nodos *workers* é suficiente para o processamento daquela tarefa.

4.4. Considerações

Como estamos tratando de escalonamento de processos, a aplicação BT do NPB executa vários *workers* (*threads*) e com outras classes de problemas. Essas classes de problemas são multiplicação e transposição de matrizes de grandes dimensões.

Na abordagem ATHA (HOSKEN, 2003) foi implementado e estabelecido um modelo de sistema que permitiu realizar processamento paralelo de forma oportunística, com tolerância a falhas. Essa abordagem ainda teve seu foco voltado para a filosofia dos sistemas implementados em grades (*grids*) computacionais. Seu objetivo principal era o de reduzir ao mínimo o custo de processamento utilizando a capacidade ociosa dos recursos conectados através da Internet.

Assim, com o intuito de validar o modelo proposto ATHA, foi implementado um protótipo e realizados vários testes com uma aplicação de quebra de chaves criptográficas, utilizando o algoritmo RC5. A mesma abordagem foi utilizada em (MENDONÇA, 2008) com o mesmo intuito de processamento paralelo de maneira oportunística. Porém, desta nova abordagem foi implementado um novo módulo ao sistema. Esse módulo proposto realiza o escalonamento junto ao gerenciador de tarefas do módulo *master*. E, baseado nas informações recebidas dos recursos dos *workers* que o módulo *master* é capaz de tomar decisões sobre essas informações recebidas, e assim enviar mais tarefas para serem executadas aos nodos escravos. Desta maneira, cada *worker* da abordagem consegue processar um *Solver*.

Nossa abordagem possui uma lista com os *Solvers* que são:

- os que não foram enviados ainda para nenhum *Worker*
- os enviados para os *Workers,* mas não retornaram ainda.

Nós chamamos essa lista de "*lista de pending*".

A aplicação BT possui "*steps*" (passos ou procedimentos) que se caracterizam por executar todos os objetos de uma etapa, ou seja, todos os N objetos do tipo *XSolver* (onde N=número de nodos). Para poder passar para o próximo *step*, ou seja, do *XSolver* para o *YSolver*, a "*lista de pending*" deve estar vazia.

O ATHA fica então pedindo *Solvers* para a aplicação BT e armazenando nesta lista. Então, esta lista é verificada se é vazia ou não, se ela não for vazia, pega o primeiro elemento da lista (que é um *Solver*), envia para um *Worker* que estiver disponível e coloca no final da lista.

Assim que um *Worker* retorna o seu *Solver* (ou seja, executou o método *step*() do *Solver*), este é removido desta *lista de pending*.

É válido lembrar que estas sincronias estão sendo feitas através dos métodos de *wait*() e *notify*() do Java.

Uma característica interessante disso é que um nodo pode falha cair e o sistema ainda irá continuar em execução. Isto porque o objeto *Solver* que foi enviado para o nodo que falhou continua na lista. No entanto, o mesmo *Solver* pode ser enviado para dois *Workers* diferentes, tendo trabalho desperdiçado, isto é, significa dizer que recursos estão sendo consumidos em vão.

Com a intenção de reestruturar a abordagem ATHA finalizamos por implementar uma nova abordagem. Com isso reformulamos todos os algoritmos de coescalonamento de processos. A figura 10 demonstra como a nova abordagem está atualmente. O nodo ADM, o qual é denominado aqui como *master* é o nodo reponsável pela submissão e recebimento das tarefas após estas serem executadas. Ele é o nodo responsável pelo gerenciamento dos processos. No nodo *master* é feita a serialização e o escalonamento dos objetos para em seguida enviá-los aos nodos escravos (*workers*). Os nodos escravos por sua vez recebem o objeto serializado e executam o processamento das tarefas pertinentes àquele processo. Isto pode ser feito em diferentes nodos paralelamente. Finalizada esta etapa, o objeto já processado é novamente serializado e o resultado reenviado ao nodo *master*.

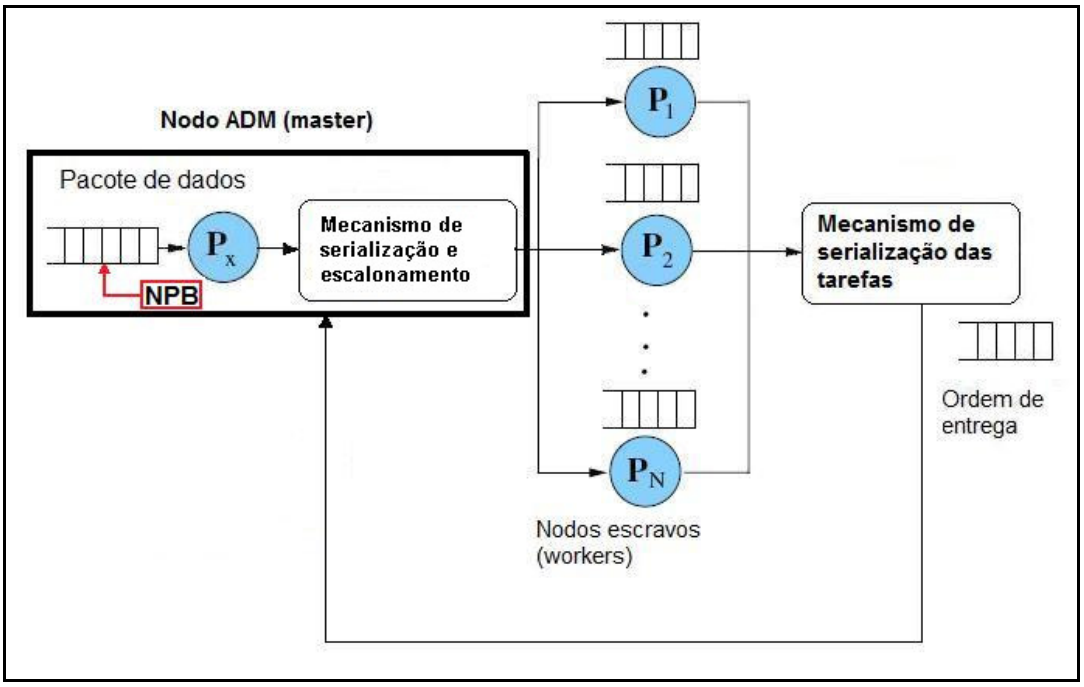


Figura 10: Abordagem com a utilização do NPB mostrando o fluxo das tarefas, o mecanismo de serialização das tarefas e sua entrega ao nodo ADM (*master*).

A partir deste momento o nodo ADM, para facilidade de compreensão, será denominado *master*, e os nodos escravos ou também como já mencionado anteriormente *slave*, serão denominados de *workers*.

4.5. Considerações sobre a proposta

Neste capítulo descreveu-se a proposta de nosso trabalho especificando o modelo proposto fazendo uma relação com os trabalhos descritos nos capítulos anteriormente objetivando especificar as principais características de funcionamento de nosso experimento.

CAPÍTULO 5: Ambiente e Resultados Experimentais

Neste capítulo apresentamos os resultados obtidos na utilização de nossa proposta em um ambiente (real e em produção) de *cluster* computacional, em conjunção a um estudo estatístico realizado para comprovar nossa abordagem.

5.1. O Ambiente Experimental

O ambiente experimental ao qual os testes foram feitos é um agregado computacional, *cluster* SMP, com um nodo *master* e seis nodos *workers*, como pode ser observado na figura 11, os quais possuem uma arquitetura *dual processor,* tendo cada nodo as caracteríticas descritas na tabela 1.

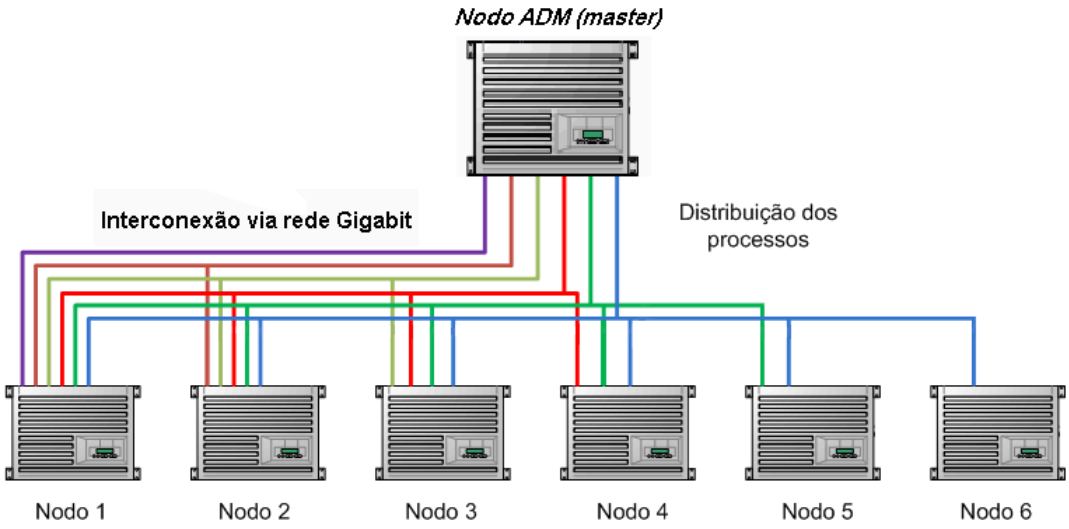


Figura 11: Configuração do ambiente experimental.

Nodo	Processador	Frequência do Processador	RAM (bytes)	Rede	*Cache L2*	S.O.
Master	Intel® Xeon®	2.66 GHz	4 GB	Gigabit Ethernet	512 KB	Linux Mandrake
Workers	Intel® Xeon®	2.66 GHz	2 GB	Gigabit Ethernet	512 KB	Linux Mandrake

Tabela 1: Características e configurações das máquinas participantes do experimento.

5.1.1. Configuração das máquinas escravas

O nodo *master* opera com o sistema operacional Linux Mandrake baseado no *kernel* versão 2.6.8.1-12mdksmp (gcc version 3.4.1 (Alpha 3.4.1-3mdk). As máquinas escravas executam uma versão do sistema operacional Linux, em sua distribuição também Mandrake igual à que está instalada no nodo *master*, a qual foi instalada via NFS. Em ambos os casos, os sistemas operacionais estão com seu suporte a SMP ativado. Outro fator importante e que cabe destaque é com relação às aplicações que estão em execução no ambiente computacional utilizado. Ele está dedicado a execução dos modelos meteorológicos (numéricos), os quais fornecem a previsão de tempo para todo o estado de Santa Catarina. Este ambiente encontra-se na EPAGRI/CIRAM (Empresa de Pesquisa Agropecuária e Extensão Rural de Santa Catarina/Centro de Informações de Recursos Ambientais e de Hidrometeorologia). Esta é uma empresa pública, responsável pela previsão meteorológica do estado de Santa Catarina, e utiliza os resultados gerados pelos modelos executados no *cluster* computacional para auxiliar a equipe de meteorologistas da empresa neste processo. Assim, serão aqui apresentados os resultados dos estudos realizados no ambiente de *cluster* da empresa.

Os experimentos foram executados igualmente para duas das seis classes da aplicação BT em estudo. Foram escolhidas dentre as seis classes (A, B, C, D, S, e W) duas de diferentes tipos de abordagem. A classe S para pequenos problemas e a classe W para execução em *workstations*, porém para problemas com granularidade um pouco maior que os da classe S. Os problemas propostos pelo *benchmark* NPB são de

multiplicação e transposição de matrizes e a ordem das matrizes é gerada aleatoriamente, sempre muito grandes, e estas operações matemáticas é que demandam muitos recursos das máquinas para executarem suas funções.

Portanto, para cada classe do *benchmark,* nossa abordagem foi executada cinco (5) vezes para cada entrada, a fim de obter uma avaliação mais precisa, amenizando resultados acidentalmente díspares. Além disso, apenas um processo é executado por máquina. Falhas de processo foram desconsideradas pelo fato de não terem ocorrido nestes experimentos, porém falhas inerentes ao processo, já que são independentes levam a propor um planejamento por blocos completamente aleatorizados.

Em nossa abordagem foram elencadas três variáveis para fazermos a coleta dos dados, e a cada execução o resultado da coleta é armazenado em arquivos de texto identificado pelo nome da variável em questão, as quais estão denominadas e descritas abaixo:

- ***stepTime_n.txt***, em que "n" é o número do *worker* envolvido. Nesse arquivo é coletado o tempo que a aplicação BT demorou para executar os cálculos dentro do *worker* "n". Esta variável, daqui por diante será denominada somente de *StepTime*.

- ***workerSerializationTime_n.txt*** em que "n" é o número do *worker* envolvido. Nesse arquivo temos o tempo de processamento e serialização do objeto para enviá-lo de volta ao *master*. Esta variável, daqui por diante será denominada somente de *SerializationTime*.

- ***workerAttendentSerializationTimer.txt*** o qual coleta o tempo que o *master* demorou para serializar o objeto. Esta variável, daqui por diante será denominada somente de *MasterTime*.

Cabe resaltar aqui que quando falamos em serialização, queremos dizer que é o tempo de seriaização somado ao tempo de envio das informações de volta ao *master*.

5.1.2. Conceitos estatísticos

A utilização dos conceitos estatísticos são importantes, pois podem indicar via utilização de determinadas ferramentas e alguns procedimentos, que os dados estatísticos coletados em função de não se apresentarem normalmente distribuídos, precisam de tratamento. O problema geral, identificado neste trabalho, diz respeito à tomada de decisão, partindo da análise dos dados, considerados normais, e na sequência a aplicação de alguns métodos estatísticos, com revisão crítica, através da aplicação da análise de variância e posteriormente aplicação das transformações por Box Cox (BOX & COX, 1964).

Desta forma, dessa seção em diante iremos utilizar alguns conceitos estatísticos importantes e aqui os definiremos para o melhor entendimento do conteúdo de nosso experimento.

a) Homocedasticidade: este conceito está relacionado com a **variância constante nos resíduos dos erros.** Como sabemos, os erros estão associados ao experimento e deveriam ser constantes com propriedades equivalentes à distribuição normal ε : N~(0, δ^2). O estimador estatístico do erro aleatório ε é conhecido como resíduo, e este, por sua vez, deve expressar as mesmas propriedades constantes esperadas nas variabilidades dos erros (MONTGOMERY & PECK, 1992).

b) Heterocedasticidade: é a forte dispersão dos dados em torno de uma reta, ou seja, é a não homocedasticidade (MONTGOMERY & PECK, 1992).

c) Intervalo de confiança: É um intervalo que contém o **verdadeiro valor** de alguma medida, cujo valor é desconhecido (MEYER, 1988)

d) Intercepto: diz respeito a inclinação (declive) de uma linha e o local onde ela cruza o eixo (intercepto). A inclinação e o intercepto definem a relação linear entre duas variáveis e podem ser usadas para estimar uma taxa média de mudança. Quanto maior a magnitude da inclinação, mais íngreme a linha e maior a taxa de mudança (MEYER, 1988).

e) Variabilidade: é a causa associada ao efeito estocástico. Existem medidas para diagnosticar estes efeitos, tais como: coeficientes de variação, variância, desvio padrão (MONTGOMERY & PECK, 1992)

f) Normalidade: distribuições que apresentam as mesmas propriedades de uma distribuição de densidade gaussiana (MONTGOMERY & PECK, 1992).

g) Independência dos dados: Correlação nula, isto é, não há alteração nos estados de uma medida quando outra propriedade (ou medida) se altera (MONTGOMERY & PECK, 1992).

Em nosso experimento, os seis nodos do ambiente computacional são nossas unidades amostrais, os quais representam as parcelas subdivididas secundárias (vide figura 27), sendo, portanto as classes S e W da aplicação BT as parcelas principais do tratamento (*SerializationTime*), e isto será explicado nas próximas seções.

5.2. Correlação entre as variáveis *StepTime, MasterTime* e *SerializationTime*

A utilização dos conceitos estatísticos são importantes, pois podem indicar uma análise essencial dos dados é a possível existência de correlação entre elas. Isso significa dizer que a variação em quaisquer das variáveis pode afetar outras de forma significativa. Dada uma parte dos dados para um único nodo, apresentamos a dispersão dos dados, conforme as figuras 12 e 13. Observa-se claramente a falta de correlação entre as variáveis, pois a sua dispersão é praticamente aleatorizada.

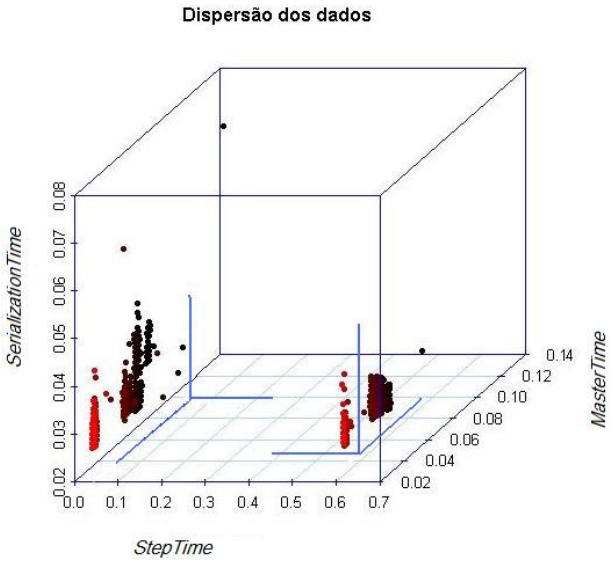


Figura 12: Dispersão dos dados no nodo 1 para as variáveis *StepTime, MasterTime e SerializationTime* (unidades em segundos).

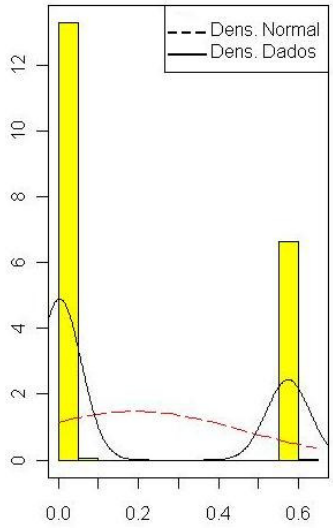


Figura 13: Dispersão dos dados para a variável StepTime (unidades em segundos).

Comparando as figuras 12 e 13 vemos que na forma de distribuição do plano, a variável *StepTime* não apresenta dados centrados, mas sim em suas extremidades. Desta forma podemos concluir que os dados são não correlacionados e não apresentaram uma possível resposta também à distribuição normal.

A tabela 2 é um demonstrativo da correlação entre as variáveis *StepTime, MasterTime e SerializationTime*, das quais se observa a inexistência de correlação entre estes tempos. Isto significa não haver, possivelmente, um modelo regressivo explicativo, que tenha uma boa confiabilidade na modelagem.

	StepTime	SerializationTime	MasterTime
StepTime	1.0000000	0.0870142	-0.2101565
SerializationTime	0.0870142	1.0000000	0.3469842
MasterTime	-0.2101565	0.3469842	1.0000000

Tabela 2: Correlação entre variáveis (unidades em segundos).

Sob este ponto de vista concluímos em verificar dois tipos de estudos estatísticos para os tempos, dos quais não há **interação ou correlação** entre tipos de tempos (*StepTime, MasterTime e SerializationTime*). A primeira análise consiste em achar um modelo que descreva o comportamento de forma significativa das classes no decorrer do aumento dos nodos; a segunda análise é o foco principal do trabalho que foi o planejamento de experimentos por parcelas subdivididas no tempo de serialização.

A figura 14 na sequência representa a distribuição dos dados referentes às variáveis *StepTime, MasterTime e SerializationTime*, e observa-se que a maior variação dos dados é em relação aos dados, do qual fizemos uma análise de ajuste do modelo por regressão a fim de diagnosticar o comportamento do tempo *StepTime* na inclusão dos nodos. O conjunto de dados *SerializationTime* é o mais relevante deste trabalho, sendo necessário o planejamento estatístico que discutiremos mais adiante (BUSSAB, 1988).

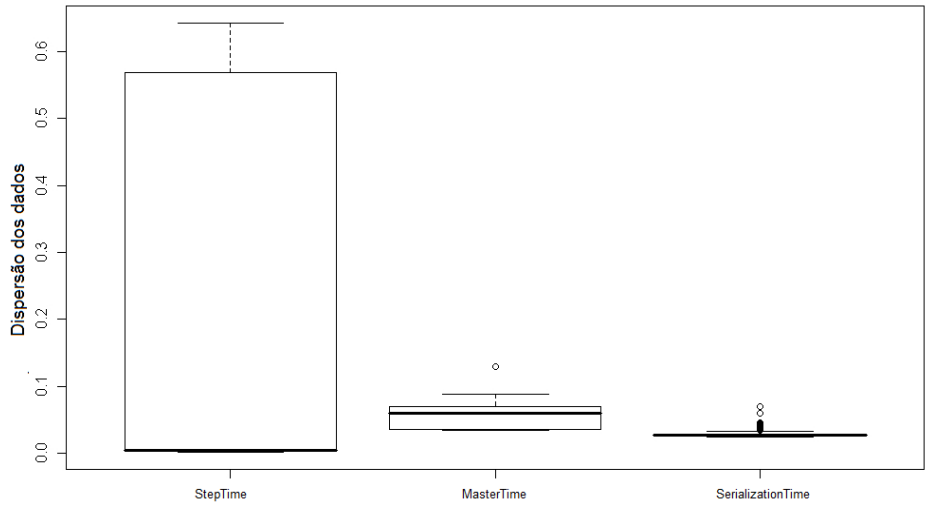


Figura 14: Distribuição dos dados referentes às variáveis *StepTime*, *MasterTime* e *SerializationTime* (unidades em segundos).

Possuímos três tipos de tempos coletados a partir das variáveis anteriormente destacadas das quais mostramos nas figuras 15, 16 e 17 a seguir.

- *StepTime*

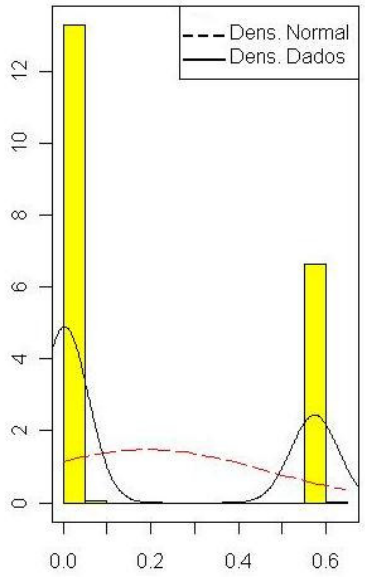


Figura 15: Variável *StepTime* (unidades em segundos).

- *MasterTime*

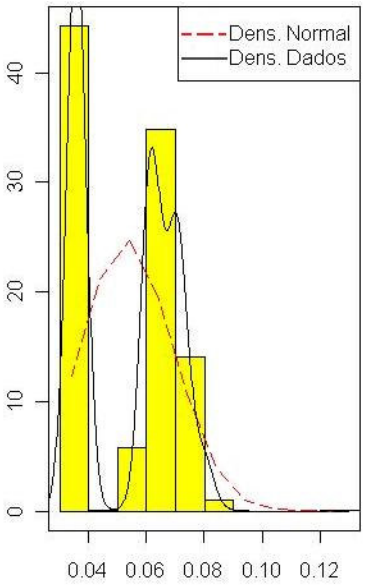


Figura 16: Variável *MasterTime* (unidades em segundos).

- *SerializationTime*

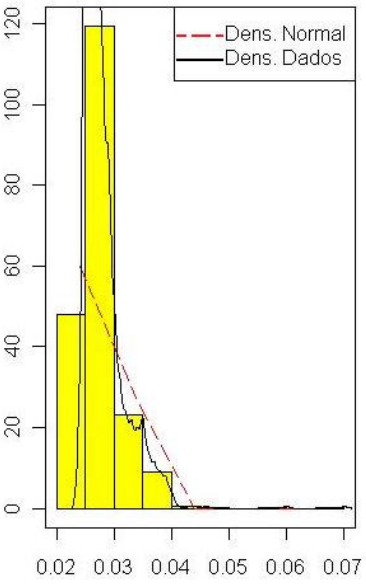


Figura 17: Variável *SerializationTime* (unidades em segundos).

De todos os dados observados acima, os mais adequados para a análise experimental são os da variável *SerializationTime*, pois apresenta o quão próximos os momentos estatísticos estão da distribuição normal.

5.3. Análise exploratória dos dados e análise regressiva

As unidades amostrais aqui utilizadas dizem respeito somente aos nodos *workers*, pois são eles que recebem as tarefas, as executam e fazem o reenvio dos dados processados ao nodo mestre. Dessa forma, o que é considerado aqui neste trabalho são somente os tratamentos dos tempos de processamento das tarefas nos nodos.

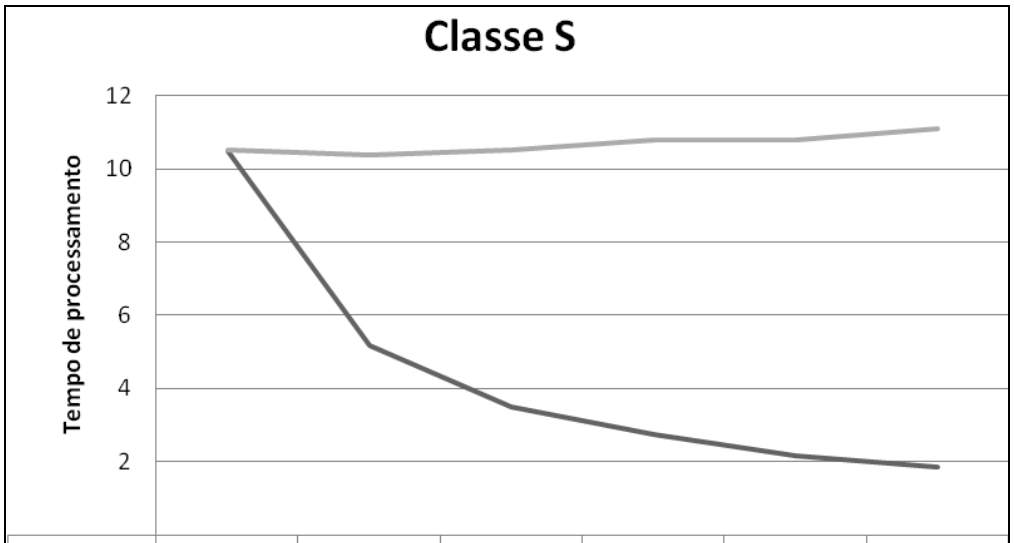
Todos os gráficos que aparecem aqui, mais algumas tabelas demonstrativas dos cálculos estatísticos e seus resultados foram feitos com o *software* estatístico R em sua versão 2.8.1 para a plataforma Windows® (R, 2008). R é um *software* livre muito utilizado para análises estatísticas computacionais e gráficas e possui distribuições para diversas plataformas de sistemas operacionais. Dentre elas podemos citar diversas plataformas UNIX, dentre estas as distribuições Linux, e as proprietárias Windows® e MacOS®.

Em primeira análise, os dados foram coletados com o resultado final do tempo total de execução dos processos nos nodos. Neste experimento, foram alocados um nodo *master* para alocação do nosso modelo, do *benchmark* utilizado, envio e recebimento das tarefas aos nodos *workers* e seis nodos *workers* para execução das tarefas da aplicação BT do *benchmark* NPB, especificamente as duas classes S e W, as quais estão na tabela 3, para cada quantil de nodo.

Quantidade de Nodos (quantil)	Media S (s)	Media W (s)	Soma S (s)	Soma W (s)
1	10,494	348,582	10,494	348,582
2	5,1865	195,533	10,373	391,066
3	3,500333	171,622	10,501	514,866
4	2,726333	99,381	10,77	397,524
5	2,1586	90,8202	10,793	454,101
6	1,848833	89,025	11,093	534,15

Tabela 3: Dados coletados e tempo total de execução das tarefas

Vemos a partir dos dados da tabela, que **as somas dos tempos** tendem a ser constantes, independentes dos quantis dos nodos. Mas as médias dos tempos finalizados **em cada um dos nodos** em relação às tarefas processadas são descendentes. Isto implica no quão rápido estes trabalhos são realizados, o que leva a crer na existência de uma correlação e algum ajuste para este comportamento exponencial em primeira análise (curva cinza escuro).

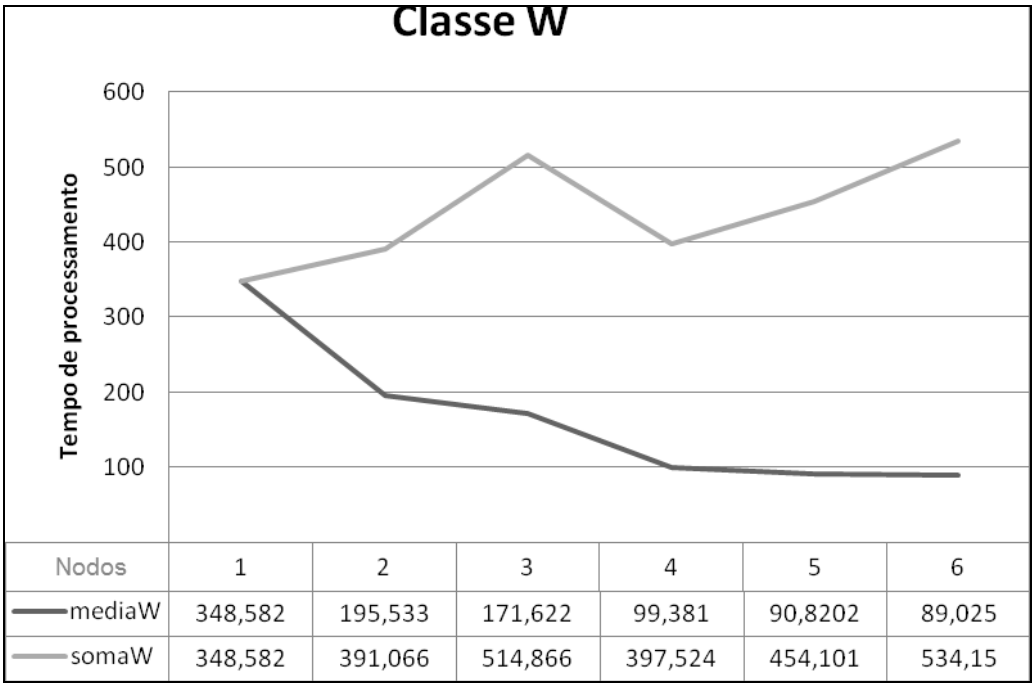


Nodos	1	2	3	4	5	6
mediaS	10,494	5,1865	3,500333333	2,726333333	2,1586	1,848833333
somaS	10,494	10,373	10,501	10,77	10,793	11,093

Figura 18: Tempo de término do processamento da classe S (unidades em segundos).

As figuras 18 e 19, nos mostram o tempo total (soma representada em verde no desenho da curva) e o tempo médio (representado em marrom na curva), do processamento por classe, sendo elas S e W.

Para os seis nodos fica claro que existe a possível queda no tempo **médio** de processamento para cada nodo em separado (vide figura 27).



Nodos	1	2	3	4	5	6
mediaW	348,582	195,533	171,622	99,381	90,8202	89,025
somaW	348,582	391,066	514,866	397,524	454,101	534,15

Figura 19: Tempo de término do processamento da classe W (unidades em segundos).

A Classe W apresentou um comportamento oscilante entre 100 unidades de tempo que vão de 350 a 550 unidades. Esta oscilação requer um estudo mais detalhado deste comportamento, uma vez que sua média mostrou-se esperadamente como as da classe S.

5.3.1. Classe S

Para o modo S foi proposta inicialmente uma aproximação linear, pois a aplicação deste requer explicar a queda do tempo médio de processamento das tarefas quando submetidas aos nodos.

A proposta implica na discussão de um modelo ajustável de primeira ordem, dado os primeiros termos da aproximação da série de Taylor de uma função exponencial (LEITHOLD, 1994):

$$e^x \approx 1 + \frac{x}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \ldots$$

$$e^x \approx \sum_{n=1}^{\infty} \frac{x^n}{n!} \Rightarrow \text{aproximadamente de primeira ordem.}$$

Isso pode ser verificado pela tabela 4, a qual testa a viabilidade da representação de um modelo linear para explicar tal comportamento. A tabela 4 nos mostra que o modelo linear não apresenta um ajuste adequado, uma vez que seu grau de explicação não passará de 70,19 % (R^2) além da heterocedasticidade (variância não constante dos resíduos) (VIEIRA, 1999).

```
Call:
lm(formula = S ~ Nodo)

Residuals:
      1       2       3       4       5       6
 2.3832 -1.4076 -1.5771 -0.8344  0.1145  1.3214

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   9.6275     1.6525   5.826  0.00432 **
Nodo         -1.5167     0.4243  -3.574  0.02329 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.775 on 4 degrees of freedom
Multiple R-squared: 0.7616,     Adjusted R-squared: 0.7019
F-statistic: 12.78 on 1 and 4 DF,  p-value: 0.02329
```

Tabela 4: Teste para validação do modelo linear para a classe S (unidades em segundos).

O teste t-Student (t-*value*) é utilizado aqui para verificar se os parâmetros do modelo polinomial β_0, β_1, $\beta_{2,}$..., β_i são importantes no modelo e, assim testar sucessivamente se são significativos.

$$y = \beta_0 + \beta_1 x + \varepsilon$$

$$H_0 : \beta_i = 0 \text{;}$$

Considera-se que o intervalo de confiança (IC):

$$(9{,}6275\,\text{s} - 1{,}6525\,\text{s};\ 9{,}6275\,\text{s} + 1{,}6525\,\text{s}) \Rightarrow (7{,}975\,;11{,}28)_{95\%}\,s$$

para **o intercepto.**

Idem para a inclinação da reta com IC:

$$(-1{,}5167\,s - 0{,}4243\,s\,;\, -1{,}567\,s + 0{,}4243\,s) \Rightarrow (-1{,}941;\, -1{,}0924)_{95\%}\, s$$

para o **coeficiente angular**.

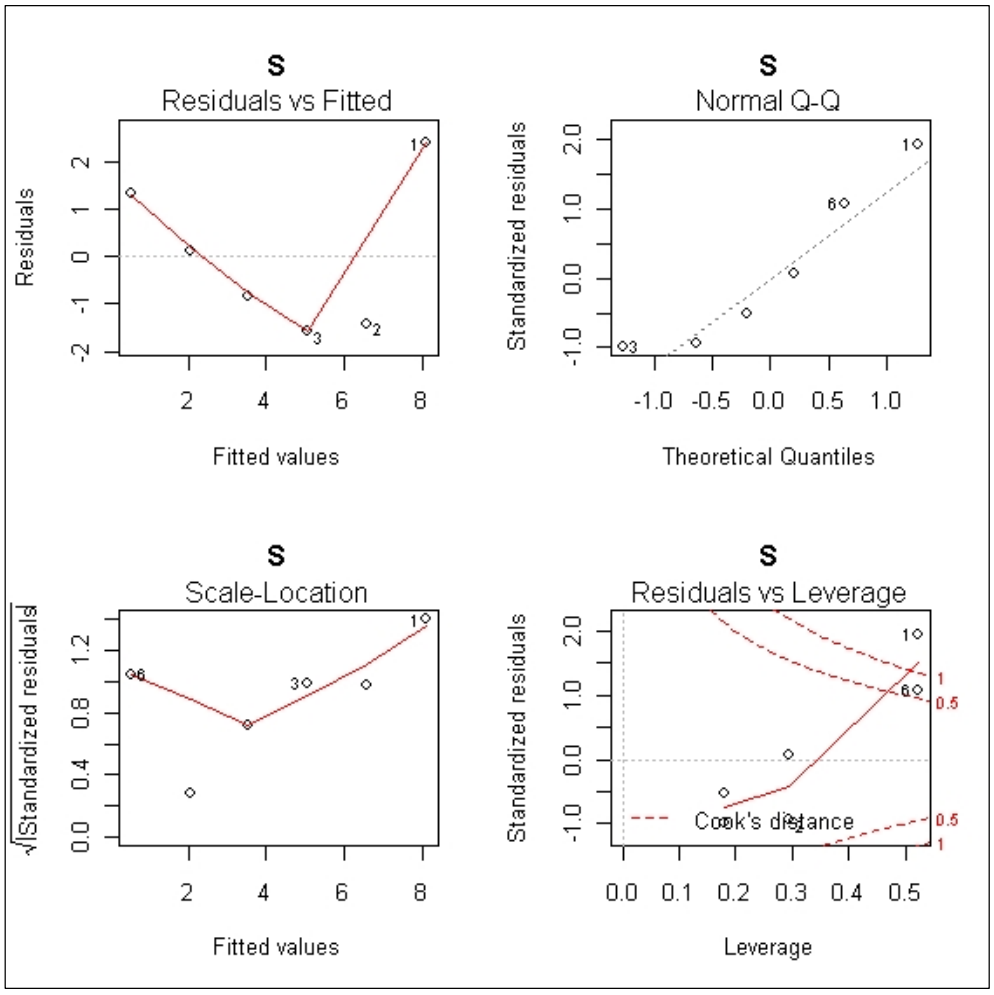


Figura 20: Validação da análise de variância (ANOVA) (unidade em segundos).

Neste caso observamos que o intercepto (β_0), quanto ao parâmetro angular (β_1) devem permanecer no modelo, dado que seus p-valor são menores do que 5% e seus IC não incluem o valor zero em seus intervalos, o que os tornam importantes ao modelo.

O problema destas análises é verificar a validade de homocedasticidade (variância constante dos resíduos), independência e normalidade dos dados. Os gráficos

da tabela 4 mostram uma tendência parabólica, não sendo, portanto, heterocedástica (VIEIRA, 1999). Isto pode ser visualizado no primeiro gráfico superior esquerdo da figura 19.

O teste demonstra que existe significância estatística para a inclusão de um modelo quadrático com 95% de confiança, grau de explicação de R^2 = 92,84%. Isso significa que com uma boa aproximação, o tempo médio de processos nos nodos cai quadraticamente. Também, existe a possibilidade do ajuste dos parâmetros β_0, β_1, e β_2 para o modelo, como mostram os testes t associados a cada parâmetro do modelo.

As propriedades pertinentes à homocedasticidade, normalidade e independência dos dados podem ser verificadas pela figura dos dados dos resíduos das figuras 20 da classe S e 23 da classe W. A figura 20 mostra o ajuste teórico em comparação com o executado.

Nota-se também nesta figura a tendência parabólica da curva (classe) dada pela tendência dos resíduos heterocedásticos quando o modelo é linear afim (função de primeira ordem).

Com a inserção de β_2 no modelo tornando-o quadrático, passamos de R^2 = 70,19% (tabela 4), para R^2 = 92,84 % (tabela 5) de capacidade de explicação deste modelo para os dados reais. Inclusive, a verificação da homocedasticidade, normalidade e independência dos dados como verificamos na tabela 5.

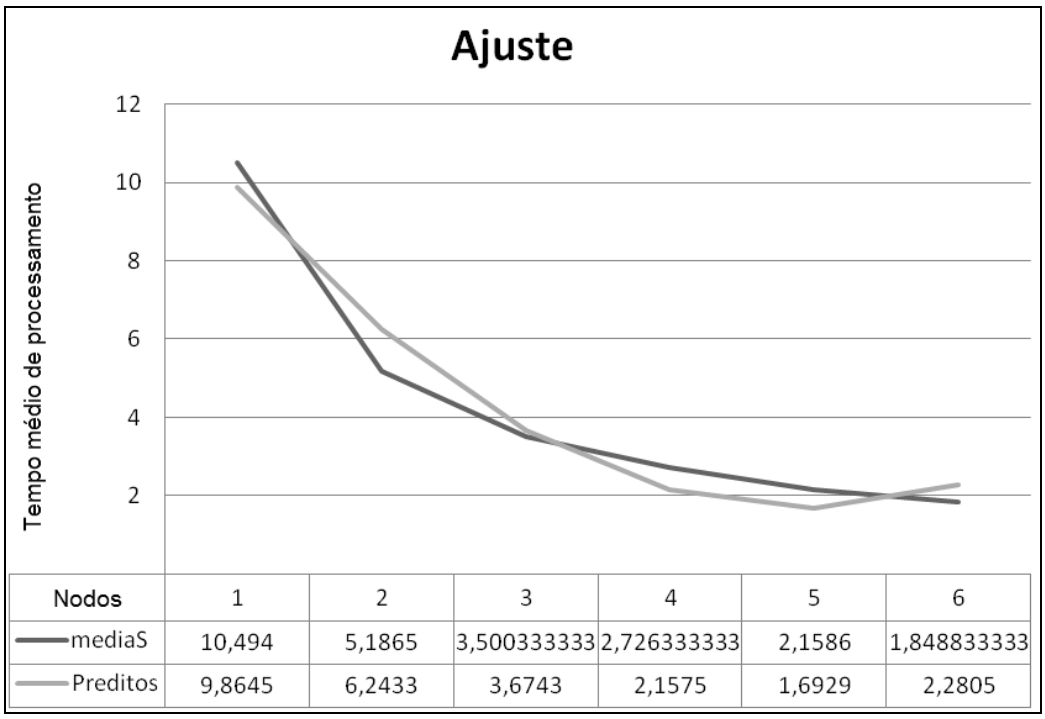
```
Call:
lm(formula = S ~ Nodo + I(Nodo^2))

Residuals:
      1       2       3       4       5       6
 0.6295 -1.0569 -0.1741  0.5685  0.4653 -0.4323

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  14.5379     1.5561   9.342  0.00260 **
Nodo         -5.1995     1.0181  -5.107  0.01452 *
I(Nodo^2)     0.5261     0.1424   3.695  0.03439 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.8699 on 3 degrees of freedom
Multiple R-squared: 0.9571,     Adjusted R-squared: 0.9284
F-statistic: 33.43 on 2 and 3 DF,  p-value: 0.0089
```

Tabela 5: Ajuste quadrático de β2 (unidade em segundos).



Nodos	1	2	3	4	5	6
mediaS	10,494	5,1865	3,500333333	2,726333333	2,1586	1,848833333
Preditos	9,8645	6,2433	3,6743	2,1575	1,6929	2,2805

Figura 21: Verificação do ajuste quadrático (unidade em segundos).

Modelo:

$$\mathbf{S_{med(predito)} = 14{,}5379 - 5{,}1995\ N + 0{,}5261\ N^2}$$

A tabela 6 confirma a proximidade do modelo.

Quantis de Nodos	mediaS	Preditos
1	10,494	9,8645
2	5,1865	6,2433
3	3,500333	3,6743
4	2,726333	2,1575
5	2,1586	1,6929
6	1,848833	2,2805

Tabela 6: Ajustes dos valores preditos em comparação com os valores reais (unidade em segundos).

A validação do modelo pode ser verificada pela figura 22, a qual valida a tabela da ANOVA (tabela 7), pois a homocedasticidade, normalidade e independência dos dados são aceitas.

Abaixo verificamos a validação da análise da variância para os **valores calculados** da classe S.

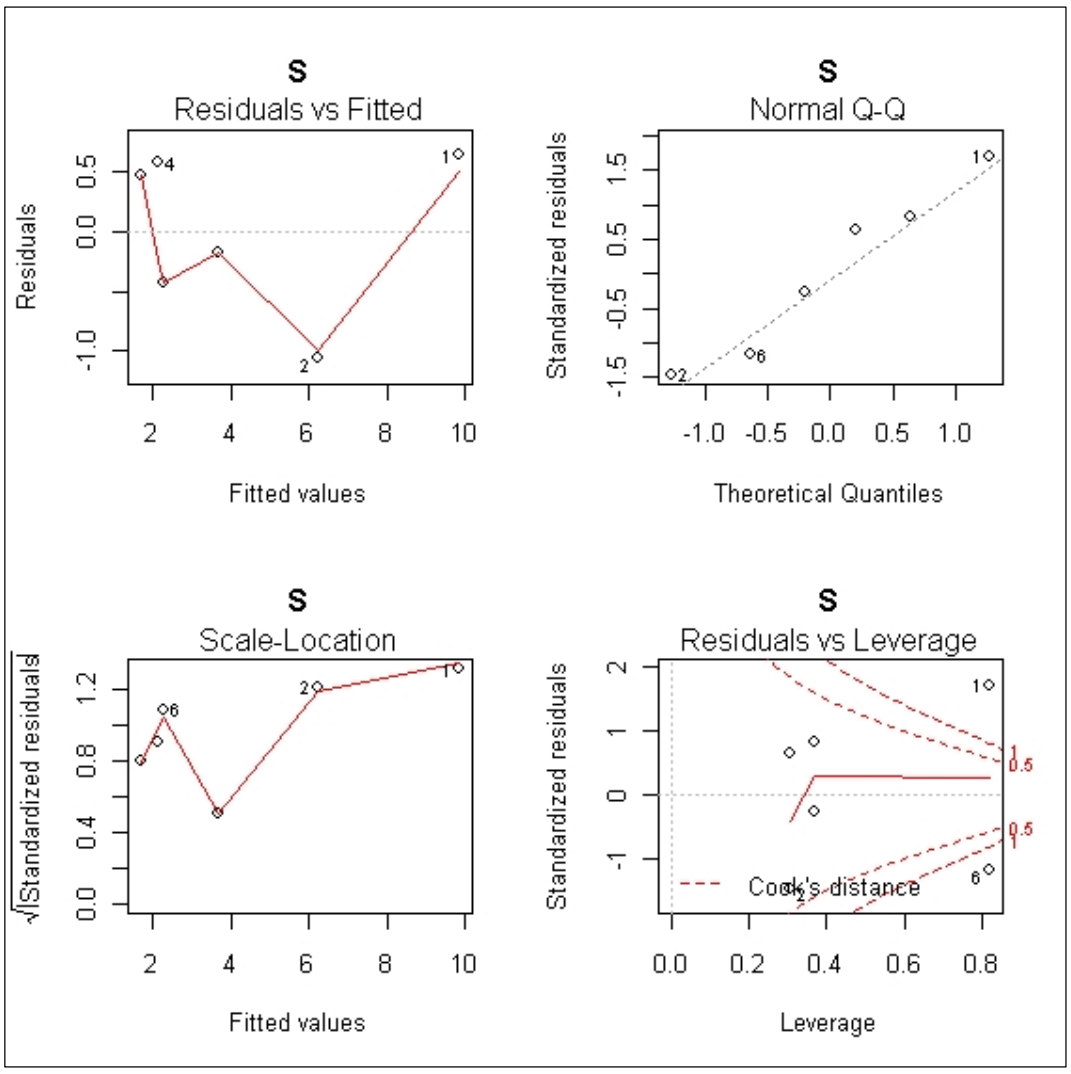


Figura 22: Validação da ANOVA para a classe S (unidade em segundos).

```
Analysis of Variance Table

Response: S
           Df Sum Sq Mean Sq F value    Pr(>F)
Nodo        1 40.255  40.255  53.197 0.005321 **
I(Nodo^2)   1 10.334  10.334  13.656 0.034387 *
Residuals   3  2.270   0.757
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Tabela 7: Análise de Variância (ANOVA) de S (unidade em segundos).

Uma informação muito importante diz respeito a Média da Soma dos Quadrados (MSQ) para os resíduos, a qual é muito menor que os termos representativos linear e quadrático (N e N^2) do modelo quadrático proposto ($\mathbf{S_{med(predito)} = 14,5379 – 5,1995\ N + 0,5261\ N^2}$) anteriormente.Vemos que MSQ residual é de 0,757 s^2 enquanto que MSQ_{Nodo} = 40,255 s^2 e I($Nodo^2$) = 10,334 s^2. Portanto, **aceita-se o ajuste**, uma vez que a maior parte da variabilidade pode ser explicada no modelo.

Como os quadrados dos erros dos resíduos são muito baixos em comparação com a soma dos quadrados dos nodos, podemos dizer que o experimento comportou-se como o esperado.

5.3.2. Classe W

Para a classe W, assim como para a classe S, propomos também uma aproximação linear em função da queda do tempo médio de processamento das tarefas nos nodos, nas mesmas técnicas utilizadas pela classe W (**tabela 17 do anexo 1**), pois também ocorre uma queda exponencial do tempo médio de processamento. Por ser polinômica também requer pelos teoremas de cálculo das séries e sequências aproximação por séries de Taylor (LEITHOLD, 1994). Dependendo do grau de precisão, o ajuste linear e o ajuste quadrático tornam-se viáveis para análise dos dados. Isso verificamos na tabela 8 que mostra a viabilidade da representação de um modelo linear para explicar esse comportamento.

```
Call:
lm(formula = W ~ Nodo)

Residuals:
      1       2       3       4       5       6
 62.457 -42.473 -18.265 -42.387  -2.829  43.495

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   334.24      45.93   7.278  0.00189 **
Nodo          -48.12      11.79  -4.080  0.01509 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 49.33 on 4 degrees of freedom
Multiple R-squared: 0.8063,     Adjusted R-squared: 0.7579
F-statistic: 16.65 on 1 and 4 DF,  p-value: 0.01509
```

Tabela 8: Teste para validação do modelo linear para a classe W (unidade em segundos).

O teste t-Student (t-*value*) é utilizado para verificação da significância dos parâmetros do modelo polinomial β_0, β_1, β_2, ..., β_n.

$$y = \beta_0 + \beta_1 x + \varepsilon$$

$$H_0 : \beta_0 = 0$$

$$H_0 : \beta_0 = \beta_1 = 0$$

Considera-se que o intervalo de confiança (IC):

$$(334{,}24\,\text{s} - 45{,}93\,\text{s};\ 334{,}24\,\text{s} + 45{,}93\,\text{s}) \Rightarrow (288{,}31;\ 380{,}17)_{95\%}\ s$$

para o **intercepto**.

Idem para a inclinação da reta com IC:

$$(-48{,}12\,\text{s} - 11{,}79\,\text{s};\ -48{,}12\,\text{s} + 11{,}79\,\text{s}) \Rightarrow (-59{,}91;\ -36{,}33)_{95\%}\ s$$

para o **coeficiente angular**.

Neste caso, observado da classe W temos que o intercepto (β_0), quanto ao parâmetro angular (β_1) **devem** permanecer no modelo, visto que seus p-valor são

menores do que 5% e seus IC não incluem o valor zero em seus intervalos. Da mesma forma, a tabela 8 demonstra o baixo grau de explicação R^2 = 75.79 % do ajuste.

O problema destas análises é verificar a validade de homocedasticidade (variância constante dos resíduos), independência e normalidade dos dados. Os gráficos da figura 23 nos mostram uma tendência parabólica. Isso é suficiente para podermos afirmar que os dados são heterocedásticos. Idem ao caso da classe S.

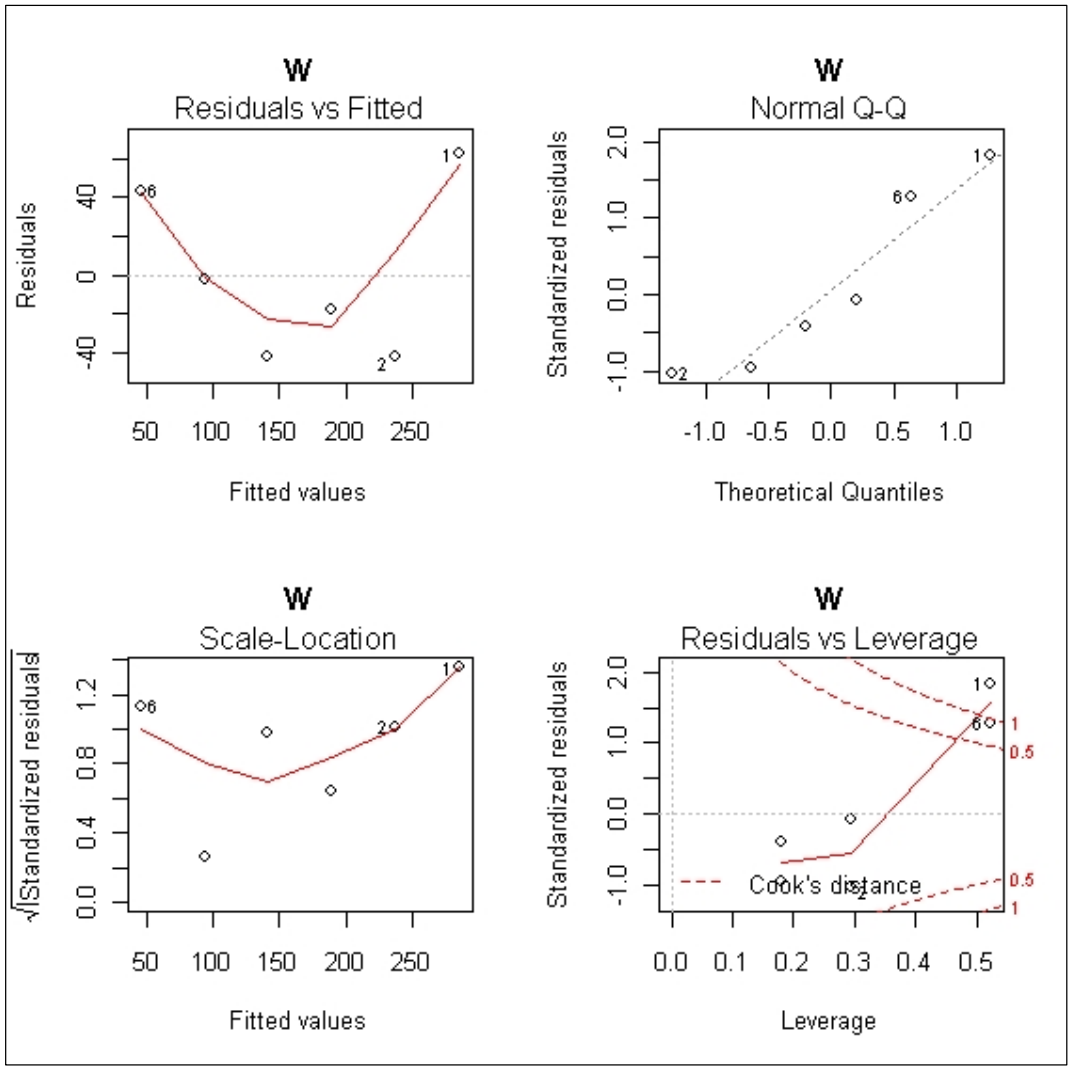


Figura 23: Validação da análise de variância (ANOVA) (unidade em segundos).

Não diferente ocorre com a classe W: tal teste demonstra que existe significância estatística para a inclusão de um modelo quadrático com 95% de confiança, grau de explicação de R^2 = 94,11%. Isso significa que, com uma boa aproximação, o tempo médio de processamento nos nodos cai quadraticamente. Existe a possibilidade do ajuste para os parâmetros β_0, β_1, e β_2 para o modelo, vide testes t associados a cada parâmetro do modelo pela tabela 9.

A tabela 9 mostra o ajuste teórico em comparação com o executado.

As propriedades pertinentes à homocedasticidade, normalidade e independência dos dados podem ser verificadas pela figura dos dados dos resíduos, uma vez que estes validam a tabela 11.

```
Call:
lm(formula = W ~ Nodo + I(Nodo^2))

Residuals:
      1       2       3       4       5       6
 13.787 -32.739  20.672  -3.450   6.906  -5.176

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  470.522     43.520  10.812  0.00169 **
Nodo        -150.328     28.472  -5.280  0.01325 *
I(Nodo^2)     14.601      3.982   3.667  0.03507 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 24.33 on 3 degrees of freedom
Multiple R-squared: 0.9647,     Adjusted R-squared: 0.9411
F-statistic: 40.96 on 2 and 3 DF,  p-value: 0.006641
```

Tabela 9: Ajuste teórico e Soma do quadrado dos erros unidade em segundos.

O modelo teórico proposto e diagnosticável como aceitável pode ser visto na tabela 9 para a classe W:

$$\mathbf{W_{med:\ (predito)} = 470{,}522\text{-}150{,}328\ N + 14{,}601\ N^2}$$

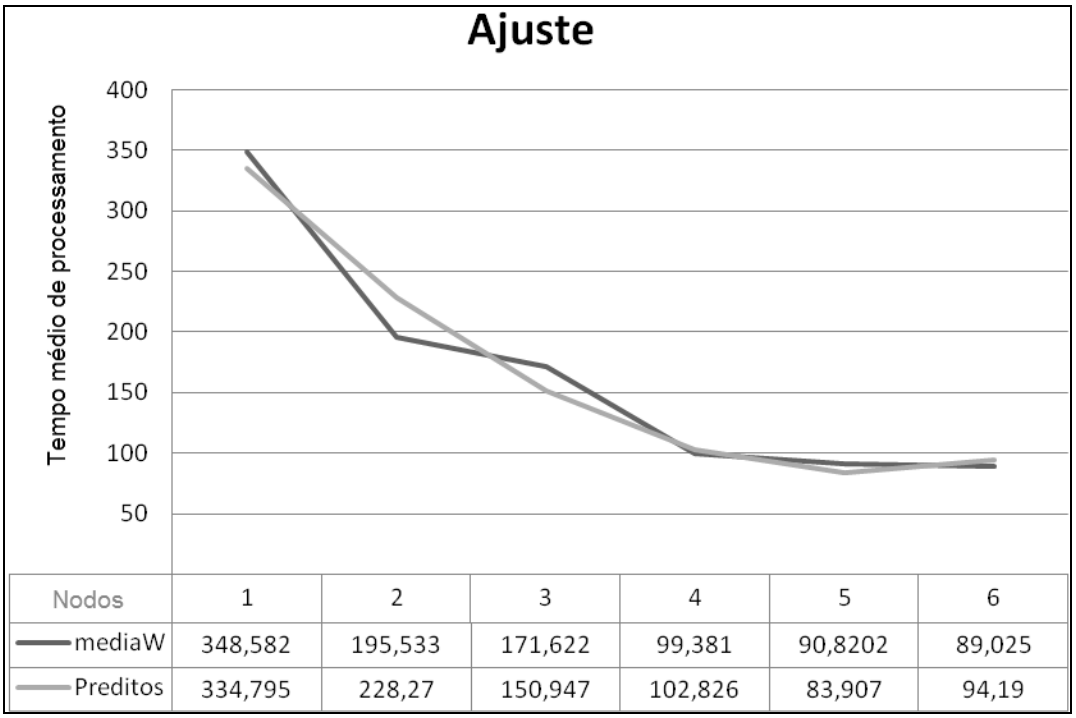


Figura 24: Verificação do ajuste quadrático (unidade em segundos)

Abaixo comprovamos a proximidade do modelo proposto anteriormente para a classe W.

Nodo	mediaW (s)	Preditos (s)
1	348,582	334,795
2	195,533	228,27
3	171,622	150,947
4	99,381	102,826
5	90,8202	83,907
6	89,025	94,19

Tabela 10: Ajuste dos valores preditos em comparação com os valores reais.

Podemos observar na tabela 10 que o ajuste parece ser explicativo, sabendo que a figura 25 valida a tabela da ANOVA.

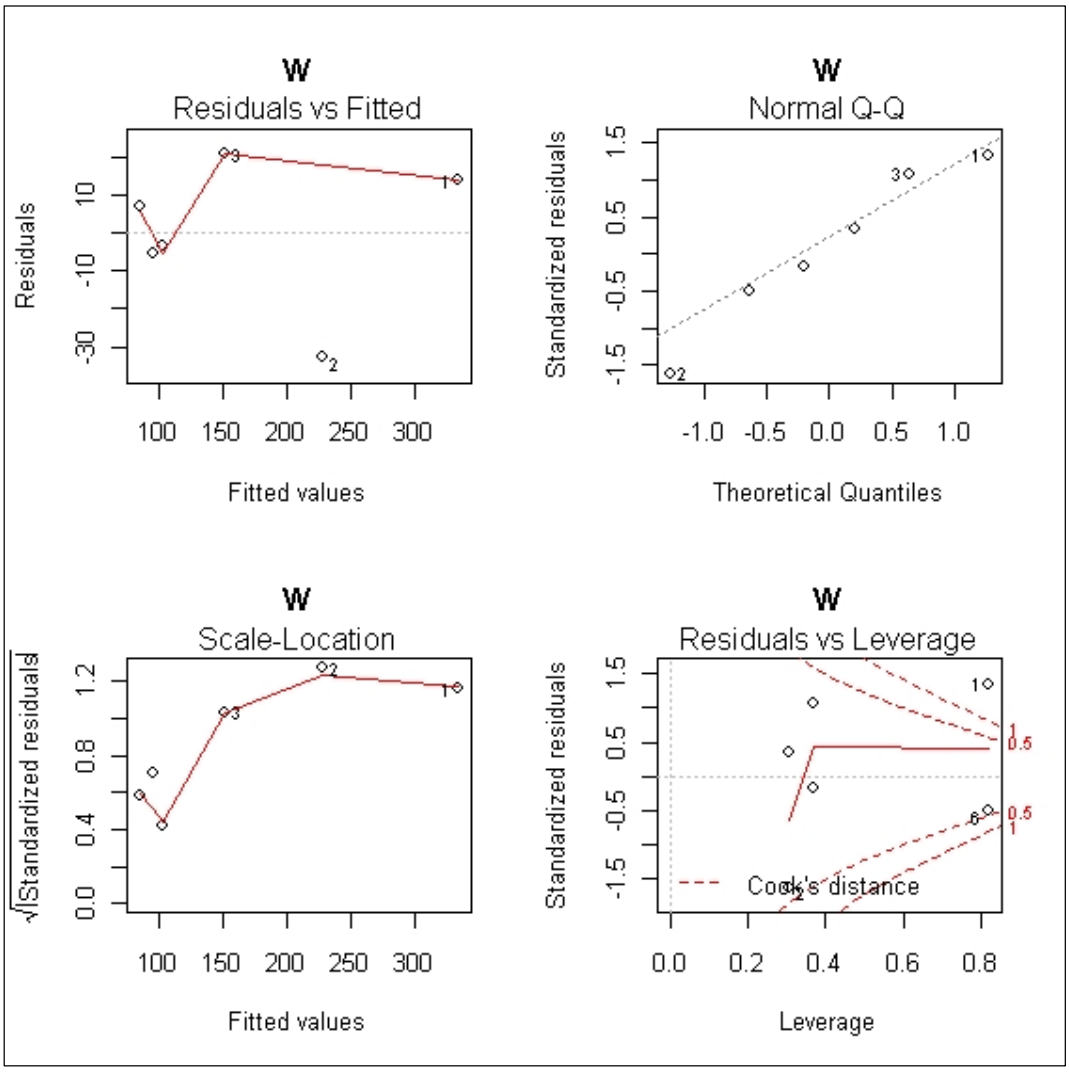


Figura 25: Validação da ANOVA para a classe W (unidade em segundos).

Podemos dizer que o experimento comportou-se como o esperado, tendo em vista que os quadrados dos erros dos resíduos são muito baixos em comparação com a soma dos quadrados dos nodos, como ocorreu com a classe S. Os valores podem ser observados na **tabela 18 no anexo 1.**

O experimento comportou-se como o esperado, pois os quadrados dos erros dos resíduos são muito baixos comparados com a soma dos quadrados dos nodos.

O que mostramos aqui é que os comportamentos da classe S e da classe W são muito semelhantes em sua modelagem. Ambas decrescem de forma quadrática, parabólica com concavidade para cima, mas **as taxas de velocidade de queda** são diferentes, e dadas por:

$$\boxed{\frac{dS}{dN} = -5{,}1995 + 1{,}0522N}$$

Mais:

$$\boxed{\frac{dW}{dN} = -150{,}328 + 29{,}202N}$$

Como as variáveis N são iguais e escrevendo $\boxed{\frac{dS}{dN} = S'}$ e $\boxed{\frac{dW}{dN} = W'}$, chegaremos à seguinte equação:

$$\boxed{S' = 0{,}03603W' + 10{,}6160}$$

o que implica dizer que a queda da classe S é mais **lenta** em **27 vezes** mais que a classe W (1/0,03603).

Isto significa dizer que, se compararmos os desempenhos das classes S e W, a média de desempenho da classe S está entre 24,39 e 58,23 vezes maior que a classe W.

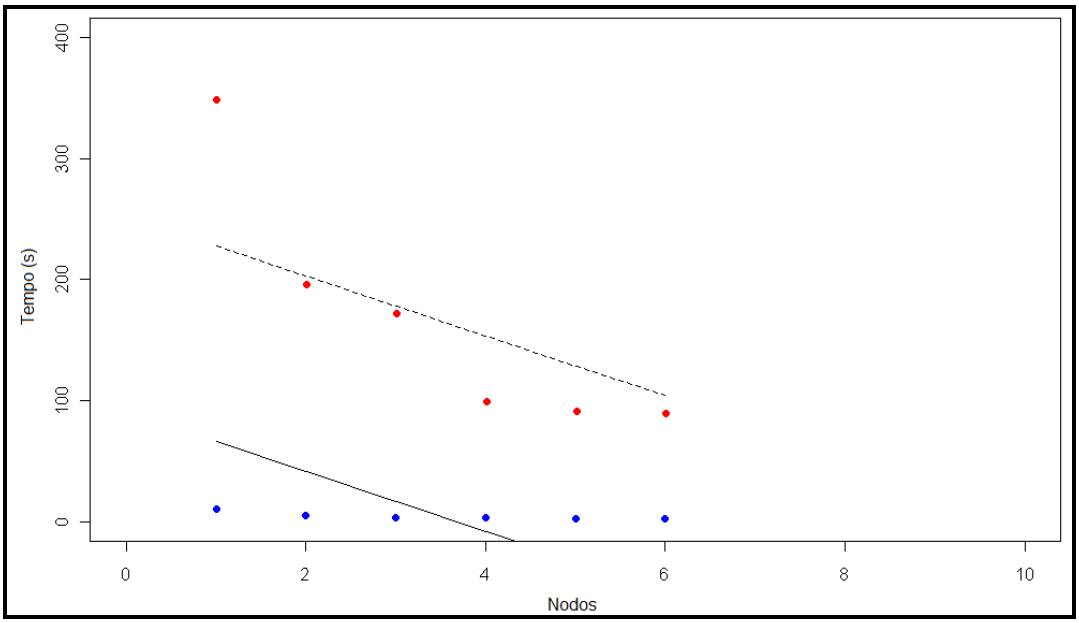


Figura 26: Avaliação do grau de diferença entre classes do *StepTime*: há diferença entre os tempos médios de processamento em ambas as classes.

Da figura 26 observamos que, até o momento, com a primeira observação, podemos dizer que há **sim** diferença no tempo médio de processamento entre as classes. Mas, por outro lado, observa-se, pelos pontos em cinza claro e em cinza escuro, que esta diferença pode, no entanto, tornar-se possivelmente não significativa para uma quantidade grande de nodos, ou à medida que o número de **nodos cresce**.

5.4. Análise de planejamento do experimento

Os resultados colhidos durante a execução da aplicação de teste foram tabulados e deles feita toda a análise estatística que descrevemos na sequência. Considerando um experimento projetado por parcelas subdivididas, temos duas classes a serem avaliadas: classe S e a classe W. Estas sendo as parcelas principais e os nodos sendo as parcelas secundárias, conforme a configuração mostrada da figura 27.

A partir deste ambiente estudamos um experimento estatístico para que pudéssemos expor de forma direta e objetiva a análise de nossa abordagem, de maneira que a mesma fosse melhor entendida pelos leitores de nosso trabalho. O resultado deste estudo culminou na tabela 11, a qual tem por objetivo expor como será feita daqui por diante a análise da nossa abordagem.

PARCELAS	PRIMÁRIA	A1 → CLASSE S A2 → CLASSE W
	SECUNDÁRIA	B1 → NODO 1 B2 → NODO 2 B3 → NODO 3 B4 → NODO 4 B5 → NODO 5 B6 → NODO 6

Tabela 11: Configuração da análise de planejamento

Nos experimentos com parcelas subdivididas existem dois tipos de tratamento em comparação: os principais e os secundários. Os tratamentos principais são designados às parcelas da maneira usual: se as parcelas são similares, sorteiam-se os tratamentos, sem qualquer restrição (experimento totalmente ao acaso); se as parcelas não são similares, primeiro organizam-se os blocos e depois sorteiam-se os tratamentos dentro de cada bloco (experimento em blocos ao acaso).

Mas começa aqui a diferença dos experimentos com parcelas subdivididas com os demais experimentos. Depois que os tratamentos principais são sorteados às parcelas, subdivide-se cada parcela em tantas subparcelas quantos sejam os tratamentos secundários; sorteiam-se então os tratamentos secundários às subparcelas de cada parcela, como mostra a figura 27. Nesta figura está o esquema de um delineamento e parcelas subdivididas, com cinco repetições de cada tratamento principal.

O planejamento do experimento consiste em uma parcela principal chamada de Classes, sendo as duas classificadas por S e W. Abaixo delas existem seis subparcelas ou parcelas secundárias, as quais são os nodos de 1 a 6 conforme a figura 27 na sequência.

Como existem diferenças entre os computadores oriundos de eventos aleatórios não controlados como a disputa por processadores, disputa pela largura de banda ocasionando *overhead* na rede, utilizamos o bloqueamento (blocos) completamente casualizado neste experimento.

Os dados dos resultados coletados escolhidos não são os da variável *StepTime*, tendo em vista que a análise exploratória de dados diagnosticou uma distribuição totalmente antissimétrica (BUSSAB, 1988), uma vez que a distribuição da variável *SerializationTime* torna-se mais conveniente.

5.4.1. Configuração do experimento de parcelas subdivididas

A figura 27 abaixo ilustra como foi elaborado o ambiente experimental utilizado para os teste de nossa abordagem.

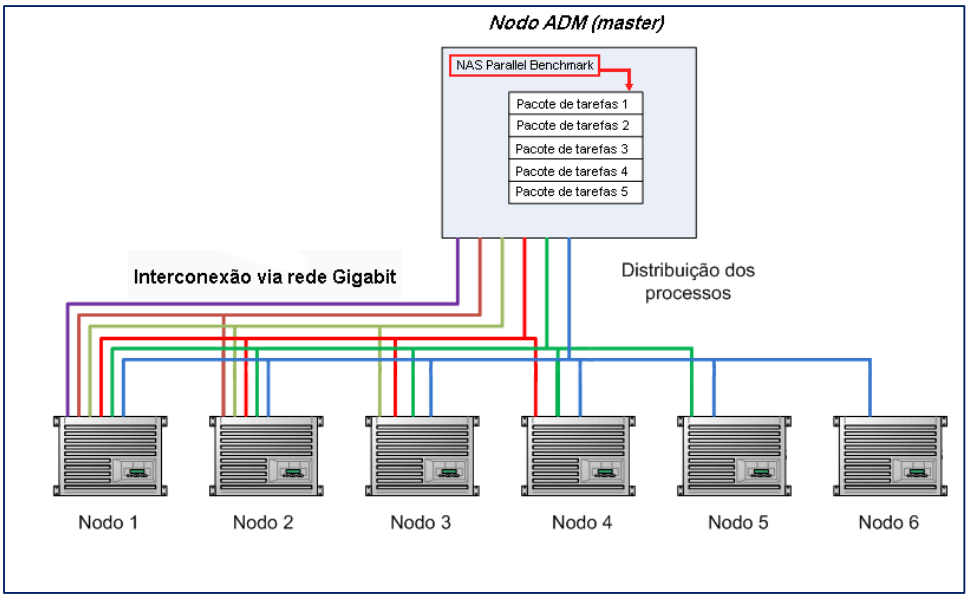


Figura 27: Configuração de análise do experimento de parcelas subdivididas

A tabela abaixo é o resultado dos tempos totais do processamento e serialização em cinco blocos, seis parcelas secundárias de B_1 a B_6 com os respectivos nodos e, finalmente as duas parcelas principais S e W.

Parcelas		Blocos					
Principal	**Secundárias**	**I (s)**	**II (s)**	**III (s)**	**IV (s)**	**V (s)**	**Total 1 (s)**
A1	B1	50,365	52,457	52,982	50,902	57,201	263,907
A1	B2	135,077	137,2	132,7	131,222	134,711	670,91
A1	B3	270,204	268,528	269,729	269,5	278,861	1356,822
A1	B4	429,279	421,458	428,065	429,833	428,142	2136,777
A1	B5	771,798	771,487	774,183	772,249	773,27	3862,987
A1	B6	1333,025	1316,917	1301,202	1322,37	1300,688	6574,202
Total 2 (s)		2989,748	2968,047	2958,861	2976,076	2972,873	14865,61
A2	B1	2,527	2,539	2,651	2,523	2,447	12,687
A2	B2	5,649	5,787	5,908	5,835	5,94	29,119
A2	B3	9,668	9,473	10,365	10,075	9,729	49,31
A2	B4	14,571	14,392	14,401	14,873	14,19	72,427
A2	B5	21,763	21,088	20,659	21,571	21,648	106,729
A2	B6	28,131	29,279	28,263	28,042	27,268	140,983
Total 3 (s)		82,309	82,558	82,247	82,919	81,222	411,255
Total 2 + Total 3 (s)		3072,057	3050,605	3041,108	3058,995	3054,095	15276,86

Tabela 12: Tabela de configuração dos experimentos

Como os dados da tabela 12 acima não apresentaram, em seus resíduos, homocedasticidades e normalidade dos dados, fez-se necessário utilizar a transformação Box Cox (BOX & COX, 1964), que neste experimento o tempo é elevado ao expoente de $Y=Y^{\wedge(2/9)}$. Essa transformação é feita para garantir as propriedades válidas de homocedasticidade , bem como a normalidade dos dados.

A tabela 13 não é referente ao tempo final de processamento t, mas sim ao tempo final de processamento no expoente 2/9, isto é, transformado.

$Y = Y^{(2/9)}$						
Parcelas		**Blocos ($s^{(2/9)}$) ou ($\sqrt[9]{s^2}$)**				
Principal	**Secundárias**	**I**	**II**	**III**	**IV**	**V**
A1	B1	2,389191	2,410896	2,416238	2,394828	2,45773
A1	B2	2,974833	2,98516	2,963119	2,955753	2,97304
A1	B3	3,470371	3,465576	3,469015	3,46836	3,494777
A1	B4	3,846389	3,830705	3,84397	3,847492	3,844123
A1	B5	4,381951	4,381558	4,384956	4,382519	4,383806
A1	B6	4,947759	4,93441	4,921263	4,938943	4,920831
A2	B1	1,228762	1,230056	1,241913	1,22833	1,220009
A2	B2	1,469281	1,477182	1,483991	1,479896	1,485773
A2	B3	1,655632	1,648152	1,681443	1,670873	1,657947
A2	B4	1,813649	1,808674	1,808925	1,821936	1,803001
A2	B5	1,982763	1,968929	1,959957	1,978862	1,98043
A2	B6	2,099139	2,11788	2,101324	2,097661	2,084654

Tabela 13: Tabela de configuração dos experimentos transformados

5.4.2. Análise de variância (ANOVA)

O modelo estatístico em parcelas subdivididas avalia o efeito das parcelas principais e secundárias. Neste experimento, o efeito do fator principal A são as classes S e W. O efeito do i-*ésimo* nodo B é dado por parcela secundária. Este experimento mostra também o efeito do i-*ésimo* bloco (cinco repetições), e este definido a seguir.

Para um experimento em Parcelas Subdivididas, o modelo estatístico (com blocos) é dado da seguinte forma:

$$y_{ijk} = \mu + \rho_i + \alpha_j + \gamma_{ij} + \beta_k + (\alpha\beta)_{jk} + \varepsilon_{ijk}$$

Onde:

μ = constante desconhecida, podendo em muitos casos ser a verdadeira média;

ρ_i = efeito do i-*ésimo* bloco, onde i=1, 2, ..., r. Em nosso caso: I, II, III, IV e V.

α_j = efeito do i-*ésimo* nível do fator A (parcela principal), onde j = 1, 2, ..., a; que em nosso caso são as classes S e W;

γ_{ij} = Resíduo a (interação entre os blocos ou repetições e o fator A ou Parcela Principal) $\gamma_{ij} \approx N(0, \sigma_\gamma^2)$;

β_k = efeito do k-*ésimo* nível do fator B (Parcela Secundária ou ainda, são os seis nodos do experimento), onde k = 1, 2, ..., b;

$(\alpha\beta)_{jk}$ = efeito da interação entre o i-*ésimo* nível do fator A e o k-*ésimo* nível do fator B;

ε_{ijk} = resíduo da interação entre o i-*ésimo* nível do fator A e o k-*ésimo* nível do fator B;

$$Aqui, \varepsilon_{ijk} \approx N(0, \sigma^2)$$

O resultado desta análise é vista na tabela 14. Atenção deve ser dada aos dados que estão transformados, pois desta forma as propriedades de homocedasticidade, normalidade e independência dos dados são garantidos conforme vimos pelas tabelas 12 e 13, que demonstram o teste de Shapiro-Wilk para a normalidade (MONTGOMERY, 1991). A tabela abaixo, 14, é validada estatisticamente pelas figuras 15 e 16 dos resíduos.

A tabela 14 mostra que existe diferença significativa entre os nodos (B) com p-valor **< 2.2^{-16} s.** Da mesma forma, existe também diferença entre as classes S e W com p-valor **< $1{,}34^{-07}$ s.** Além disso, é importante salientar na existência do efeito da interação entre o i-*ésimo* nível do fator A (níveis S e W) com o k-*ésimo* nível do fator B

(níveis nodos: 1, 2, 3, 4, 5 e 6), sendo importante analisar também os efeitos dos níveis de A dentro de B e os níveis de B dentro de A, isto é:

- Níveis de A dentro de B: a interrelação dos níveis das classes dentro dos nodos interagindo de forma significativa. Ou seja, o efeito das classes interferem nos níveis dos nodos (no processamento das informações).
- Níveis de B dentro de A: a interrelação dos níveis dos nodos dentro das classes interagindo de forma significativa. Ou seja, o efeito dos nodos interferem nos níveis das classes (no processamento das informações).

Um importante resultado é que os blocos não afetam o experimento, uma vez que o planejamento por blocos completamente casualizados não é significante estatisticamente. Por outro lado, para garantir a segurança na resposta dos dados, temos que o bloqueamento, além de opcional, é conveniente por vários motivos, dos quais podemos salientar:

Análise com os dados Transformados por BOX COX				$Y' = Y^{(2/9)}$	
Causas de Variação	**GL**	**Soma dos Quadrados** $((s^{(2/9)})^2$ **ou** $(\sqrt[9]{s^2})^2$	**Quadrados Médios** $((s^{(2/9)})^2$ **ou** $(\sqrt[9]{s^2})^2)$	**F calculado**	**p-valor (%)**
Blocos	4	0,000325	8,11E-05	0,29798	0,866088
Classes (A)	1	57,6090	57,6090	211.583	$\mathbf{1{,}34^{-07}}$
Erro **a**	4	0,001	0,000272		
Parcelas	9	57,61032			
Tratamento Nodos (B)	5	19,2537	3,8507	38.507	$\mathbf{< 2.2^{-16}}$
AB	5	4,5301	0,906	27.719	$\mathbf{< 2.2^{-16}}$
Erro **b**	40	0,0056	0,0001		
Total	59	81,39972			

Tabela 14: ANOVA com dados Transformados por BOX COX

A tabela 14 segue o formalismo da tabela da ANOVA (vide anexo 1, figura 18), isto é, como o experimento foi constituído por ***efeitos fixos*** faz com que a esperança dos quadrados médios sigam essa tabela, conforme o uso do método de Kicks (MONTGOMERY, 1991). Dado que tal efeito é fixo, este experimento é válido ***somente e somente para*** a comparação entre médias, seguindo a seguinte hipótese:

- $H_0 = \bar{A}_1 = \bar{A}_2 = ... = \bar{A}_n$
- H_1: pelo menos um $\bar{A}$ diferente

Observamos pela tabela 14 que na ANOVA a variabilidade associada aos resíduos de **a** e de **b** são baixos quando comparados com os quadrados médios (QM) das classes e das interações de (AB). Isto quer dizer que, o $QM_{\text{resíduo } \mathbf{a}} = 0{,}000272$ $(\mathbf{s}^{(2/9)})^2$ e $QM_{\text{ resíduo } \mathbf{b}} = 0{,}0001$ $(\mathbf{s}^{(2/9)})^2$; que comparados com $QM_{\mathbf{A}} = 57{,}60$ $(\mathbf{s}^{(2/9)2})^2$, $QM_{\mathbf{B}} = 3{,}85$ $(\mathbf{s}^{(2/9)})^2$ e $QM_{\mathbf{AB}} = 0{,}906$ $(\mathbf{s}^{(2/9)})^2$, realmente a variabilidade pode ser explicada.

O teste da normalidade para os resíduos mostra que tais dados não podem ser representados por uma distribuição normal, pois com o p-valor = $1{,}356 \times 10^{-7} < 5\%$, rejeita-se a hipótese de possível normalidade dos dados.

```
> shapiro.test(residuos)

        Shapiro-Wilk normality test

data:  residuos
W = 0.8001, p-value = 1.356e-07
```

Tabela 15: Teste de Shapiro-Wilk para testar a validação da normalidade dos resíduos antes da transformação BOX COX (unidade em segundos)

Quando os dados são transformados por Box Cox (BOX & COX, 1964), existe a possibilidade dos dados serem representados por distribuição normal, e torna-se viável dado p-valor = 0,3335 > 5%, aceitando a hipótese de possível normalidade.

```
>  shapiro.test(residuos)

        Shapiro-Wilk normality test

data:  residuos
W = 0.9775, p-value = 0.3335
```

Tabela 16: Teste de Shapiro-Wilk para testar a validação da normalidade dos resíduos <u>após</u> transformação BOX COX (unidade em segundos).

Para que possamos validar a ANOVA, a figura 28 é de suma importância. Ela apresenta a homocedasticidade e normalidade adquirida, conforme o gráfico localizado na parte esquerda superior da figura 28 para homocedasticidade e gráfico localizado na parte esquerda inferior da figura 28 para a normalidade. Já na parte superior direita desta mesma figura, o gráfico Box Plot mostra a dispersão dos dados, a qual se mostra bastante compacta, ou seja, os tempos de processamento estão bem próximos.

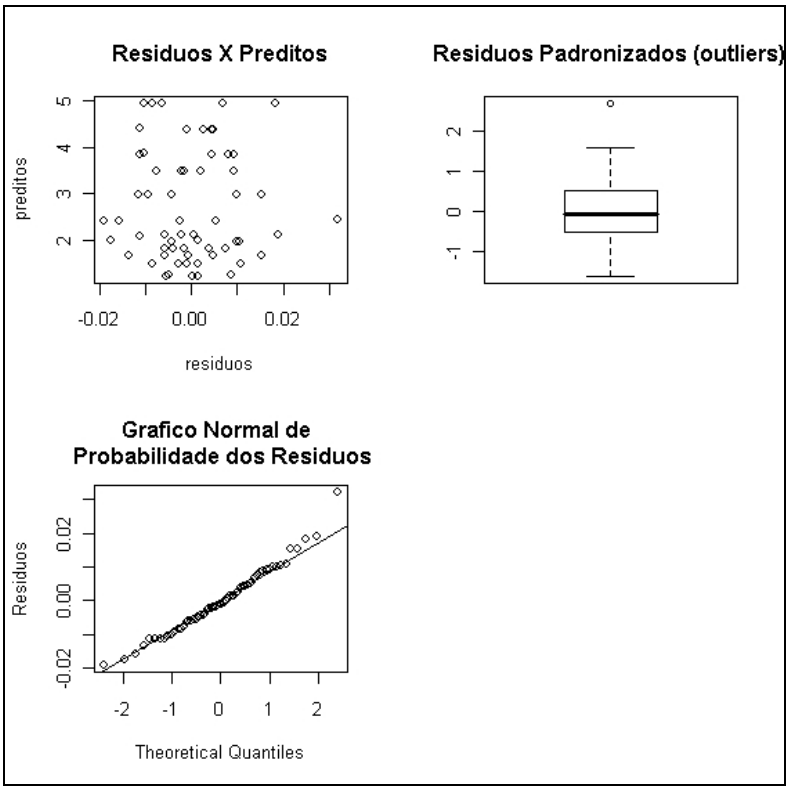


Figura 28: Gráficos dos Resíduos

5.5. Análise de variância – Estudo gráfico

O gráfico abaixo apresenta duas figuras, sendo que a da esquerda representa o grau de significância da diferença entre o i-*ésimo* efeito dos nodos nas suas respectivas classes. Já a segunda figura, a da direita, representa o grau de significância da diferença entre o i-*ésimo* efeito das classes nos seus respectivos nodos. Observa-se a diferença estatística entre os nodos e as classes, bem como a interação AB (classes e nodos) dado pela tabela de análise de variância com p-valor **$< 2.2^{-16}$ %.**

De fato, as classes respondem de maneira diferente para com os nodos. Cada classe (S e W) corresponde a uma parcela com propriedades como visto na análise de regressão anteriormente. Isto significa dizer que a classe S possui uma velocidade de processamento maior que a classe W, mas, por outro lado, a rapidez de queda no desempenho na velocidade de processamento da classe S é 27 vezes menor que a classe W. Esta diferença pode ser observada tanto pela tabela da ANOVA, tabela 14, quanto na figura 28.

Vemos aqui, com 95% de confiança, os **p-valor < 5%** rejeitam as seguintes hipóteses:

1) $H_0 : \tilde{A}_1 = \tilde{A}_2$

2) $H_0 : \tilde{B}_1 = \tilde{B}_3 = \tilde{B}_4 = \tilde{B}_5 = \tilde{B}_6$

3) H_0 : a interação AB é não significativa

Isto significa dizer que:

a) Existe pelo menos uma classe diferente em média.

b) Existe pelo menos um nodo diferente em média. Porém, em termos de arquitetura de *hardware,* todos são semelhantes. Mas, em função da velocidade de processamento, nada podemos afirmar, tendo em vista que o ambiente de *cluster* computacional está em produção e serve aos meteorologistas da empresa executarem seus modelos numéricos. Assim,

não há como medir a parcela da carga de trabalho que os processadores destinam a execução destes modelos e a parcela destinada à execução do nosso experimento no exato momento da leitura dos tempos de processamento.

c) A interação entre as classes e os nodos é significativa.

Os gráficos da figura 29 representam o quão distintos são as classes e os nodos. Vemos que, para os nodos, as classes apresentam retas que não se cruzam, e por isso são em média diferentes. Podemos dizer o mesmo para as classes quando comparados aos nodos, pois os as retas dos mesmos não se cruzam em nenhuma das cinco repetições do experimento.

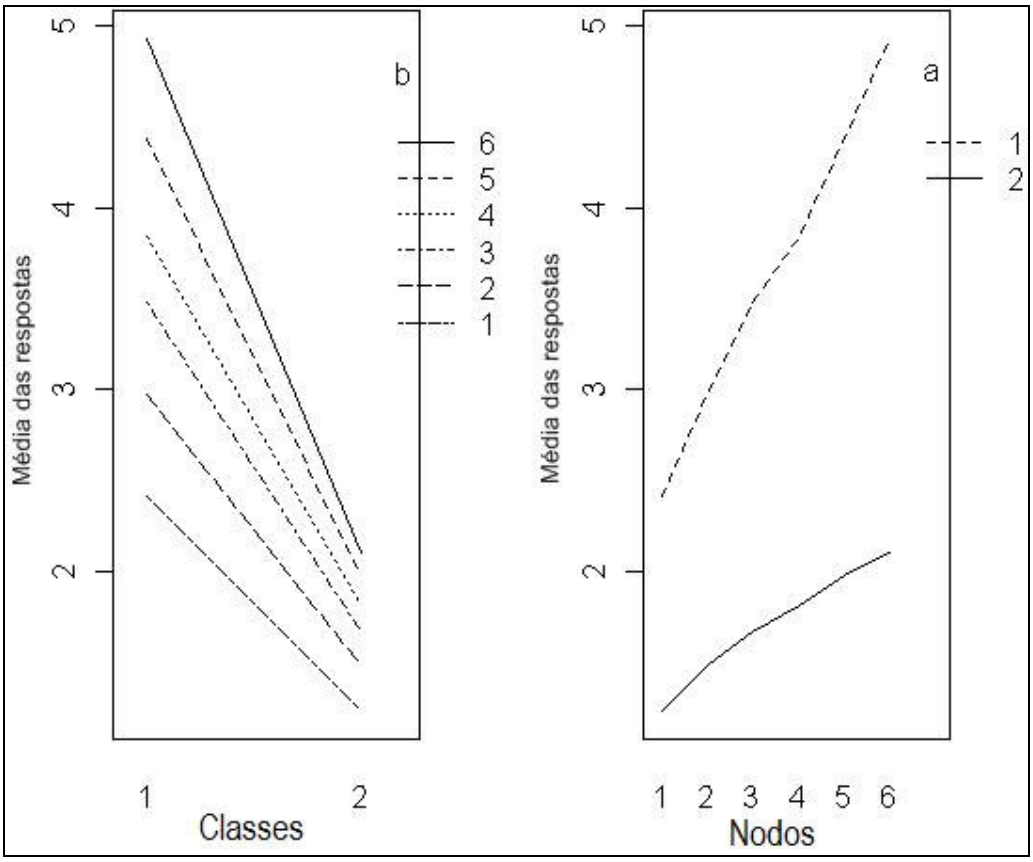
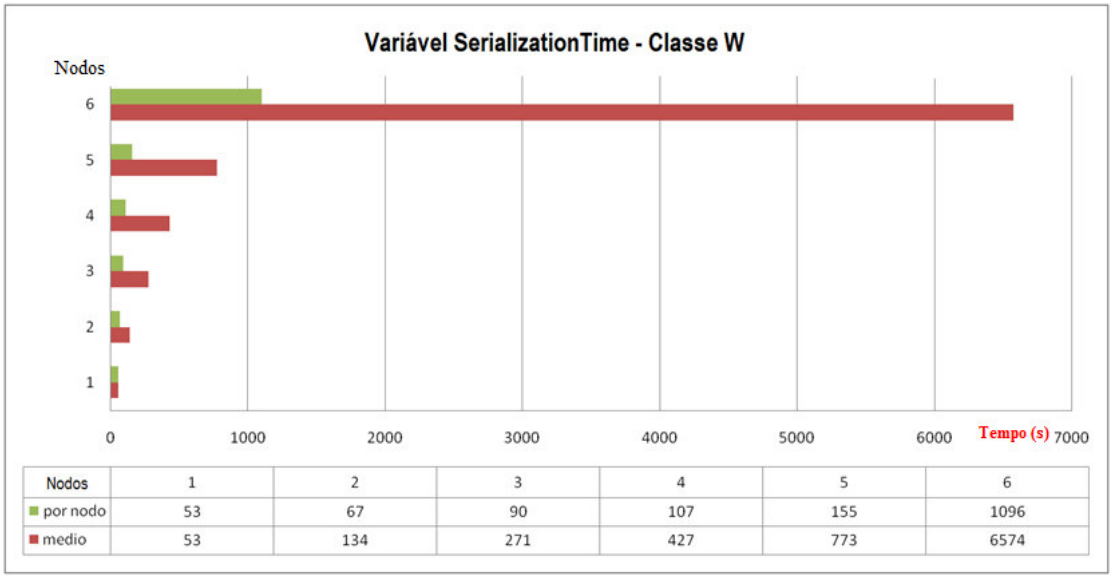


Figura 29: Nodos: representa o grau de significância da diferença entre o i-ésimo efeito dos nodos nas suas respectivas classes. Classes: representa o grau de significância da diferença entre o i-ésimo efeito das classes nos seus respectivos nodos.

O gráfico da figura 30 demonstra os resultados obtidos dos cálculos realizados com a variável *SerializationTime* da classe W. Podemos notar que há um crescimento de forma exponencial do tempo de resposta. Isto pode ser explicado em função do aumento do tráfego na rede e pela disputa pelos processadores, uma vez que o *cluster* computacional está em ambiente de produção sendo utilizado para o processamento dos modelos meteorológicos da empresa.



Nodos	1	2	3	4	5	6
por nodo	53	67	90	107	155	1096
medio	53	134	271	427	773	6574

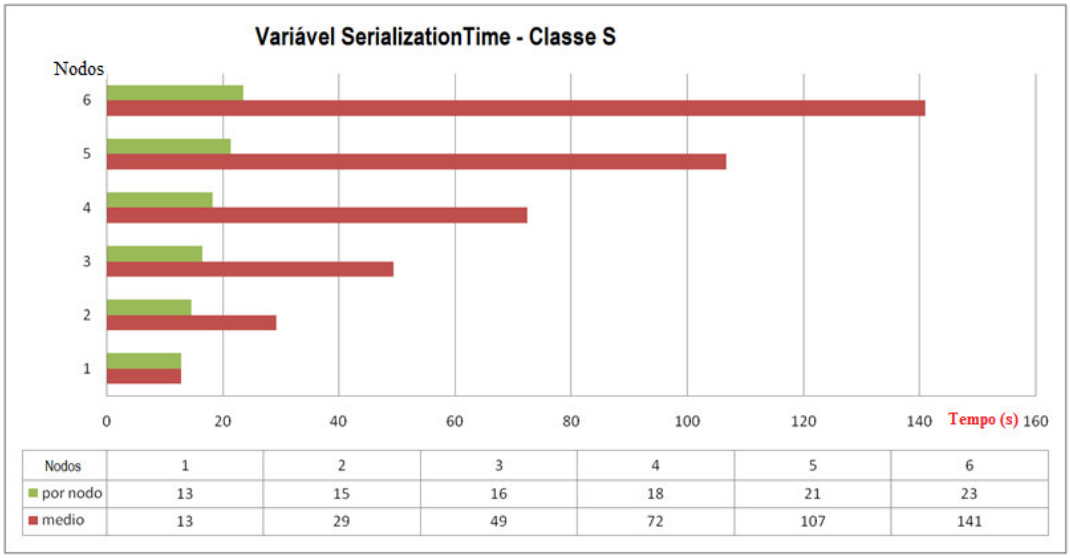
Figura 30: Variável SerializationTime da classe W

Outro fator a ser observado neste gráfico está relacionado ao tempo de resposta que passa a ordem de crescimento de 1000 segundos na utilização de seis nodos em função da implementação da classe W, onde os cálculos de transposição e multiplicação das matrizes é maior. Nessa classe a ordem das matrizes é maior que a ordem das matrizes multiplicadas e transpostas na classe S, e isso explica a regressão vista anteriormente (quadrática). A ordem das matrizes utilizadas na classe W é 24 X 24 X 24. Outro fator que aqui é relevante em relação ao tempo de resposta é o número de processadores utilizados para a execução das tarefas. Na figura 31, logo abaixo do gráfico desta figura, podemos ver a quantidade de nodos que foram utilizados no experimento; números sequenciais de 1 a 6. Então, quando a aplicação faz o envio das

tarefas aos nodos, esta já sabe a quantidade de processadores que serão utilizados para realização do trabalho de execução.

Como os nodos possuem dois processadores cada (*dual processor*), conforme já foi mencionado no capítulo 4 e 5, no momento em que é iniciada a aplicação e enviada a classe para processamento, esta já deve saber quantos processadores serão utilizados para a execução que foi proposta. Sendo assim, cada número da tabela abaixo do gráfico (números de 1 a 6) representa dois destes processadores. Em tempo, a leitura destes números faz-se de duas maneiras: (a) a primeira diz respeito ao número dos nodos aos quais as tarefas são enviadas e, (b) a segunda está relacionada ao número de processadores que são utilizados para a execução das tarefas, sendo sempre múltiplos de dois, isto é, 2, 4, 6, 8, 10 e 12.

O gráfico abaixo nos mostra os resultados obtidos dos cálculos realizados com a variável *SerializationTime* da classe S. Nele observa-se, também um crescimento de forma exponencial do tempo de resposta. Isto pode ser explicado em função do aumento do tráfego na rede e pela disputa pelos processadores, tendo em vista que o *cluster* computacional está em ambiente de produção sendo utilizado para o processamento dos modelos meteorológicos da empresa.



Nodos	1	2	3	4	5	6
por nodo	13	15	16	18	21	23
medio	13	29	49	72	107	141

Figura 31: Variável SerializationTime da classe S

Fatores que devem ser levados em conta para a classe S são os mesmos fatores que consideramos para a classe W, porém com algumas diferenças, a saber: a ordem das matrizes utilizadas na classe S é 12 X 12 X 12; e a quantidade de nodos é a mesma, assim como o número total de processadores.

O tempo de resposta desta classe chega à ordem de crescimento de um pouco mais de 20 segundos utilizando-se seis nodos.

5.6. Considerações dos Resultados Experimentais

Este capítulo mostrou como foi montada a configuração do ambiente experimental, como foi construído o experimento de análise estatística, bem como toda a metodologia utilizada para avaliar os resultados. Todos os resultados foram avaliados em função das três variáveis utillizadas no experimento para extrair os resultados das medições: *StepTime, MasterTime e SerializationTime*. Com a utilização dos métodos estatísticos aplicados sobre os dados das medições feitas pudemos verificar que estes métodos podem ser afetados pela suposição inadequada de normalidade dos dados. Com isso, uma boa avaliação, aplicando alguns procedimentos de tranformação utilizados para ajustar estes dados coletados, tornou-se possível criar mecanismos para evitar que conclusões duvidosas fossem assumidas como verdadeiras.

A análise inicial dos dados do experimento indicou que, quando foram analisados após sua coleta (apenas dados puros), muito pouco poderíamos inferir. Porém, percebemos que estes dados necessitavam de um tratamento para podermos validar o experimento. Empregando métodos estatísticos a estes dados utilizando análise de variância e transformação Box Cox comprovamos nossa averiguação em relação ao desempenho do ambiente de *cluster* computacional e as duas classes de problemas.

No próximo capítulo apresentamos nossas considerações finais e onde sujerimos encaminhamentos para trabalhos futuros.

CAPÍTULO 6: Conclusões e Trabalhos Futuros

Este capítulo finaliza nosso trabalho, onde apresentamos algumas considerações finais, as dificuldades encontradas e concluímos com propostas de trabalhos futuros com o intuito de continuidade para essa abordagem.

6.1. Considerações Finais

Nossa proposta foi avaliar o desempenho de um *cluster* SMP explorando o processamento paralelo distribuído em conjunto a uma abordagem de co-escalonamento. Nossa abordagem nos mostrou e comprovou, com o protótipo implementado e utilizado, as diversas técnicas utilizadas no mundo computacional no que tange a arquitetura baseada em *clusters* computacionais, em nosso caso os COTS (*commodity-off-the-shelf*). Além de serem uma excelente alternativa na obtenção de computação de alto desempenho, a relação preço/desempenho também se torna bastante satisfatória. Este tipo de arquitetura possibilita as organizações e as instituições de pesquisa um excelente ambiente computacional de investigação e trabalho. Mais do que isso, a arquitetura é bastante escalável e sempre que necessária, a adição de mais nodos é facilmente aplicável. Outro fator importante a pensar, diz respeito à heterogeneidade do ambiente, onde se pode agregar novas tecnologias em função da escalabilidade do ambiente tornando-o computacionalmente mais poderoso. Um ponto fraco que pode ser observado está relacionado à taxa de transmissão e latência. Em outras palavras, o atraso da rede, também conhecido como *overhead*, é um dos quesitos que, se não bem dimensionado, pode deixar muito a desejar, inviabilizando todo o trabalho.

Os resultados experimentais nos mostraram, como esperado, que se não bem adequada ao ambiente computacional, as aplicações não utilizaram com maior eficiência todos os recursos disponíveis de uma arquitetura de alto desempenho, principalmente no que diz respeito às arquiteturas multiprocessadas e *multi-cores*.

No primeiro momento de nossa avaliação, em função da necessidade de melhor observação de nosso experimento, afinal estávamos utilizando um ambiente real e em uma aproximação da realidade também de ambientes da computação de alto desempenho, utilizamos métodos estatísticos, neste caso em particular o método de regressão linear para avaliar as diferenças entre as classes da aplicação BT do NPB em função da variável *StepTime*, que é a variável que possui o tempo de execução de uma tarefa dentro dos nodos *workers*. Essas diferenças foram constatadas, sendo este o ponto de partida para toda nossa comprovação.

Em seguida fizemos uma nova análise de nossa aplicação utilizando análise de experimentos estatísticos. Como esperado, houve a comprovação da mesma diferenciação, esta agora em função da variável *SerializationTime*, a qual é a variável que possui o tempo de processamento e serialização das tarefas nos *workers* mais o tempo de envio destas informações de volta ao nodo *master*.

6.2. Conclusões

O ambiente utilizado para os testes foi o da EPAGRI/CIRAM aonde a proposta de coescalamento se mostrou atrativa, posto que a classe S atingiu um desempenho da ordem de 29,39 a 58,23 vezes mais rápido que a classe W. Com o aumento do número de nodos a classe S apresentou uma queda no tempo de processamento 27 vezes mais lenta do que a classe W. Isto quer dizer que se aumentando o número de nodos provavelmente não haverá diferenciação no tempo de processamento entre as classes analisadas.

Em relação aos dados, foi observado que além da **diferença entre as classes e entre os nodos** também existe diferença significativa entre nodos e classes, o que nos leva a crer na possibilidade desta interação influir no tempo de processamento.

De acordo com a variável *StepTime*, o tempo de processamento segue em aproximação quadrática. Esta aproximação é de segunda ordem, uma vez que para um modelo aproximado de uma queda exponencial apresentou um p-valor de 0,03507 e R^2 de 94,11%.

6.3. Trabalhos futuros

Investigações mais aprofundadas acerca dos métodos utilizados nesta dissertação podem ser feitas com o intuito de buscar melhorias de desempenho computacional e numérico. Outras investigações podem ser conduzidas na análise dos resultados, objetivando a compreensão de outros comportamentos dos casos estudados.

Para a continuidade de nossa abordagem, destacamos as seguintes sugestões:

- Comparação com outras aplicações de alto desempenho e que possuam outras áreas de atuação concomitante a atividades operacionais e de produção mais intensas, se possível em escala maior que esta por nós utilizada a fim de quantificar seus benefícios, caso se aplique;

- Utilização de outras tecnologias de interconexão, ou seja, diferente da rede Gigabit aqui utilizada, para avaliação da diferenciação de taxas de transmissão entre elas, caso existam;

- Agregar ao experimento outras classes da aplicação BT em conjunto, se possível, com outro ambiente computacional e contrastar seu desempenho com os valores aqui conseguidos;

- Agregar ao experimento outros *benchmarks* para considerar, em conjunto com o co-escalonamento de processos, o balanceamento de cargas e se possível em fatias de tempo (*time slices*) pré-estabelecidas;

- Adicionar ao experimento bibliotecas de comunicação e passagem de mensagem como OpenMP, MPI, MPICH, dentre outras, e avaliar o ganho de desempenho do ambiente computacional;

- Verificar a possível causa da degradação da taxa de transmissão com a utilização de RMI;

- Análise da dinâmica do tempo de processamento para milhares de nodos: Assintoticidade e caso limítrofe;

- Análise de sobrevivência no estudo de falhas e dados censurados com a inclusão (ou não) de programas "sujos";
- Inserção de outras classes diferentes da W e da S, incluindo-as no planejamento de experimentos.

Bibliografia

(ÅHLANDER & FRISK, 2006) ÅHLANDER, A.; FRISK, T.; “Introduction to radar signal processing”, Course Material, April 2006.

(AHMAD & HALSALL, 1993) AHMAD, R.; HALSALL, F. Interconnecting high-speed LANs and backbones. IEEE Network, v. 7, n. 5, p. 36-43, Sep. 1993.

(ALAM et al, 2006) ALAM, S.R. et al. Characterization of Scientific Workloads on Systems with Multi-core Processors. In: IEEE INTERNATIONAL SYMPOSIUM ON WORKLOAD CHARACTERIZATION (IISWC), 2006, Proceeding... IEEE International Symposium on Workload Characterization (IISWC), 2006, pp. 11 oct. 2006.

(ANDERSON et al, 1995) ANDERSON, T. at al. A Case for NOW (Networks of Workstations). IEEE Micro, pages 54–64, February 1995.

(BAILEY et al, 1991) BAILEY, D. H. et al. The nas parallel benchmarks. International Journal of Supercomputer Applications, v. 5, n. 3, p. 63-73, 1991.

(BELSLEY; KUH & WELSH, 1980) BELSLEY, D.A.; KUH, E.; WELSH, R.E. Regression diagnostics. New York: John Wiley & Sons, 1980.

(BOX & COX, 1964) BOX, G.E.P. and COX, D.R. An Analysis of Transformations. Journal of Royal Statistical Society. B, 39, 211-252, 1964.

(BUNTINAS, MERCIER & GROPP, 2006) BUNTINAS, D., MERCIER, G. and GROPP, W. Design and evaluation of Nemesis, a scalable, low-latency, message-passing communication subsystem. In: SIXTH IEEE INTERNATIONAL SYMPOSIUM ON CLUSTER COMPUTING AND THE GRID (CCGRID’06), 2006, Singapure. Proceeding… Sixth IEEE International Symposium On Cluster Computing And The Grid (Ccgrid’06), 2006, pp. 521–530, 2006.

(BUSSAB, 1988) BUSSAB, W. O. Análise de variância e de regressão. São Paulo: Atual, 1988.

(CASAVANT & KUHL, 1988) CASAVANT, T. L.; KUHL, J. G. A taxonomy of scheduling in general-purpose distributed computing systems. IEEE Trans. Software Eng., IEEE Press, Piscataway, NJ, USA, v. 14, n. 2, p. 141 – 154, 1988. ISSN 0098-5589.

(CHAI, HARTONO & PANDA, 2006) CHAI, Lei; HARTONO, A.; PANDA, D. K. Designing high performance and scalable MPI intra-node communication support for clusters. In: IEEE INTERNATIONAL CONFERENCE ON CLUSTER COMPUTING, 2006. Proceedings... IEEE International Conference on Cluster Computing, 2006, pp.1-10.

(COULOURIS, DOLLIMORE & KINDBERG, 2005) COULOURIS, G.; DOLLIMORE, J.; KINDBERG, T. Distributed systems: concepts and design. 4 ed. Boston: Addison-Wesley Longman Publishing, 2005.

(CULLER & SINGH, 1999) CULLER, David E.; SINGH, Jaswinder Pal. Parallel computer architecture: a hardware/software approach. San Francisco: M. Kaufmann, 1999. 1025p. ISBN 1558603433.

(DA SILVA, CARVALHO & HRUSCHKA, 2004) DA SILVA, Fabricio A.B., CARVALHO, Silvia, HRUSCHKA, Eduardo R. A scheduling algorithm for running Bag-of-Tasks data mining applications on the grid. In: CONGRÈS EURO-PAR 2004 PARALLEL PROCESSING, 2004, Pisa, Italy. Proceedings... Euro-Par 2004, ago.-set. 2004.

(DANTAS, 1997) DANTAS, M. A. R. Efficient Scheduling of Parallel Applications on Workstation Cluster. PhD Thesis in Computer Science. Electronics and Computer Science Department. University of Southampton, UK, 1997.185f.

(DANTAS, 2005) DANTAS, Mario A. R. Computação distribuída de alto desempenho: redes, clusters e grids computacionais. Rio de Janeiro: Axcel Books, 2005, pp. 278 ISBN 8573232404.

(DASGUPTA, 1990) DASGUPTA, S. A hierarchical taxonomic system for computer architectures. IEEE Computer. v. 23, n. 3, p. 64-74, March 1990.

(DONGARRA, 2003) DONGARRA, J. et al. Sourcebook of Parallel Computing. Morgan Kaufmann, 2003.

(DRAPER and SMITH, 1981) DRAPER, Norman Richard; SMITH, Harry. Applied regression analysis. 2. ed. New York: J. Wiley, 1981, c1966. 709p. (Wiley series in probability and mathematical statistics) ISBN 0471029955 (enc.)

(FARAJ & YUAN, 2002) FARAJ, Ahmad; Xin YUAN. Communication characteristics in the NAS parallel benchmarks. In: FOURTEENTH IASTED INTERNATIONAL CONFERENCE ON PARALLEL AND DISTRIBUTED COMPUTING AND SYSTEMS, November 4-6., 2002, Cambridge. Proceedings... Cambridge: Fourteenth Iasted International Conference On Parallel And Distributed Computing And Systems, 2002, pp.729-734.

(FORTUNA, 2000) FORTUNA, A. O. Técnicas computacionais para dinâmica dos fluídos: conceitos básicos e aplicações. 1. ed. São Paulo: EDUSP, 2000. ISBN-13: 9788531405266.

(FOSTER, 1995) FOSTER, I. Designing and Building Parallel Programs. 1a. ed. [S.l.]: Addison-Wesley Publishing Company Inc., 1995.

(FOX et al, 1995) FOX, Armando et al. Cluster-based scalable network services. In: 16TH ACM SYMPOSIUM ON OPERATING SYSTEMS PRINCIPLES, 1995, Canada. Proceeding…Canada, 1995, pgs. 78-91.

(FRUMKIN et al, 2002) FRUMKIN, M. et al. Implementation of the NAS Parallel Benchmarks in Java NAS Technical Report NAS-02-009, NASA Ames Research Center, Moffett Field, CA, 2002.

(GONÇALVES e FLEMMING, 1994) GONÇALVES, Mirian Buss; FLEMMING, Diva Marília. Calculo C: funções vetoriais, integrais curvilíneas, integrais de superfície. 3. ed. São Paulo: Makron, 2000. 425p. ISBN 8534609551.

(GOODMAN, 1983) GOODMAN, J. R. Using cache memory to reduce processor-memory traffic. In ISCA '83. In: OF THE 10TH ANNUAL INTERNATIONAL SYMPOSIUM ON COMPUTER ARCHITECTURE, 1983, Los Alamitos, CA, USA. Proceedings…Los Alamitos: IEEE Computer Society Press, 1983, pp.124–131, ISBN 0-89791-101-6.

(GRAMA, 2003) GRAMA, A. et al. Introduction to Parallel Computing. Addison-Wesley, 2 ed., 2003.

(HENNESSY & PATTERSON, 2003) HENNESSY, John L.; PATTERSON, David A. Computer architecture a quantitative approach. 3rd. ed. Amsterdam: M. Kaufmann, 2003. 760p. ISBN 1558607242.

(HIMENO, 2008) HIMENO Benchmark. Disponível em: <http://accc.riken.jp/HPC/HimenoBMT/contest_e.html>. Acesso em: 11 jul. 2008.

(HOSKEN, 2003) HOSKEN, A. A parallel processing environment for opportunist on the internet. Brasília, DF: UnB, 2003. Originalmente apresentada como dissertação de mestrado, Universidade de Brasília, 2003.

(HOSKEN & DANTAS, 2004) HOSKEN, A.; DANTAS, M.A.R. The ATHA environment: Experience with a user friendly environment for opportunistic computing. In: 18TH INTERNATIONAL SYMPOSIUM ON HIGH PERFORMANCE COMPUTING SYSTEMS AND APPLICATIONS – HPCS, 2004, Winnipeg, Manitoba, Canada. Proceedings… Winnipeg, 2004.

(JIN, FRUMKIN & YAN, 2008) JIN, H.; FRUMKIN, M.; YAN, J. The OpenMP Implementation of NAS Parallel Benchmarks and Its Performance. NAS Division, NASA Ames Research Center, Moffett Field, CA 94035-1000. Disponível em: <http://www.nas.nasa.gov/Resources/Software/ware/npb.html>. Acesso em: 25 jul. 2008.

(KENNEDY & McKINLEY, 1993) KENNEDY, K.; McKINLEY, K. S. Maximizing loop parallelism and improving data locality via loop fusion and distribution. In: WORKSHOP ON LANGUAGES AND COMPILERS FOR PARALLEL PROCESSING, 6, Aug.1993, Portland, OR. Proceedings… Portland: Workshop On Languages And Compilers For Parallel Processing, 1993, pp.301-20.

(KREMIEN and KRAMER, 1992) KREMIEN, O., KRAMER, J. Methodical Analysis of Adaptative Load Sharing Algorithms IEEE Trans. On Parallel and Distributed Systems, vol. 3, no. 6, Nov. 1992 pp. 747-760.

(LAM/MPI, 2008) MPICH2. Disponível em: <http://www.mcs.anl.gov/research/projects/mpich2/index.php>. Acesso em: 25 set. 2008.

(LAM/MPI, 2008) LAM/MPI. Disponível em: <http://www.lam-mpi.org>. Acesso em: 25 set. 2008.

(LAPESD, 2008) LAPESD: Laboratório de Pesquisa em Sistemas Distribuídos, UFSC, INE, CTC, 2008. Disponível em: <http://www.lapesd.inf.ufsc.br/>. Acesso em: 08 dez. 2008.

(LEITHOLD, 1994) LEITHOLD, Louis. O calculo com geometria analítica. 3. ed. São Paulo: Harbra, c1994. 2v.

(LINPACK, 2008) LINPACK Benchmark-Parallel. Disponível em: <http://www.netlib.org/linpack>. Acesso em 20 out. 2008.

(LU, 2004) LU, Qingda et al. Applying MPI derived data types to the NAS Benchmarks: A case study. In: OF THE 2004 INTERNATIONAL CONFERENCE ON PARALLEL PROCESSING WORKSHOPS, Aug. 2004, Montreal, Quebec, Canada. Proceedings…Montreal: Of The 2004 International Conference On Parallel Processing Workshops, 2004, pp. 538-545.

(MAK & LUNDSTROM, 1990) MAK, V. W.; LUNDSTROM, S. F. Predicting performance of parallel computations. IEEE Transactions on Parallel and Distributed Systems. v.1, n.3, p. 257-270, July 1990.

(MENDONÇA, 2008) MENDONÇA, R.P. Uma proposta de coescalonamento adaptativo para um ambiente de computação oportunista de recursos distribuídos. Florianópolis, SC: UFSC, 2008. Originalmente apresentada como dissertação de mestrado, Universidade Federal de Santa Catarina, 2008.

(MENDONÇA & DANTAS, 2008) MENDONÇA, R.P., DANTAS, M.A.R. A coscheduling approach for an opportunistic software environment. In: 14TH INTERNATIONAL EUROPEAN CONFERENCE ON PARALLEL AND DISTRIBUTED COMPUTING, EURO-PAR, 2008, Las Palmas de Gran Canaria, España. Proceedings... Las Palmas de Gran Canaria: 14th International European Conference on Parallel and Distributed Computing, Euro-Par, 2008.

(MENDONÇA, PIFFER & DANTAS, 2008) MENDONÇA, R.P.; PIFFER, M.M.; VIEGAS, D.R. e M.A.R. Dantas. Uma Abordagem de Coescalonamento Adaptativa para Ambientes de Computação Oportunista de Configurações Multiprocessadas. In: CLEI 2008 XXXIV Conferencia Latinoamericana de Informática, 2008, Santa Fé. Anais do CLEI 2008 XXXIV Conferencia Latinoamericana de Informática, 2008.

(MEYER, 1988) MEYER, P. L. Probabilidade – Aplicações à Estatística LTC- Livros Técnicos e Científicos R.J., 1988. 444 p. ISBN: 8521602944.

(MONTGOMERY, 1991) MONTGOMERY, Douglas C. Design and analysis of experiments. 3ed. New York: John Wiley, 1991. 642 p.

(MONTGOMERY and PECK, 1992) MONTGOMERY, Douglas C.; PECK, Elizabeth A. Introduction to linear regression analysis. 2. ed. New York: J. Wiley, c1992. 527 p. (Wiley series in probability and mathematical statistics. Applied probability and statistics) ISBN 0471533874

(NAS, 2008) NAS Parallel Benchmarks, Disponível em: <http://www.nas.nasa.gov/NAS/NPB>. Acesso em: 18 jun. 2008.

(NETER, WASSERMAN and KUTNER, 1990) NETER, J.; WASSERMAN, W.; KUTNER, M.H. Applied Linear Statistical Models. 3. ed. Illinois: Richard D. Irwin, Inc., 1990.

(PARK et al, 2003) PARK, Kyung Lang et al. Dynamic topology selection for high performance MPI in the grid environments. In: CONGRÈS RECENT ADVANCES IN PARALLEL VIRTUAL MACHINE AND MESSAGE PASSING INTERFACE, 2003, Venice, Italy. Proceedings… Congrès Recent advances in parallel virtual machine and message passing interface, 2003, vol. 2840, of Lecture Notes in Computer Science, pages 595–602, 2003.

(PINTO, 2004) PINTO, A. S. R. Abordagem de Escalonamento Dinâmico de Tarefas Baseada em Sistemas Classificadores. Florianópolis, SC: UFSC, 2004. Originalmente apresentada como dissertação de mestrado, Universidade Federal de Santa Catarina, 2004.

(POURREZA & GRAHAM, 2007) POURREZA, Hossein; GRAHAM, Peter. On the Programming Impact of Multi-core, Multi-Processor Nodes in MPI Clusters. In: 21ST INTERNATIONAL SYMPOSIUM ON HIGH PERFORMANCE COMPUTING SYSTEMS AND APPLICATIONS, 2007, Washington, DC, USA. Proceedings…21st International Symposium On High Performance Computing Systems And Applications, 2007.

(R, 2008) R Development Core Team. R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing. Vienna, Austria, 2008. ISBN: 3-900051-07-0. Disponível em: <http://www.R-project.org>. Acesso em 04 fev. 2008.

(SHIVARATRI, 1992) SHIVARATRI et al. Load Distributing for Locally Distributed Systems, IEEE Computer, vol. 25, 1992, pp. 33-44.

(SODAN, 2005) SODAN, A. C. Loosely coordinated coscheduling in the context of other approaches for dynamic job scheduling: a survey. Concurrency and computation: Practice & Experience, Canada, v.17 n.15, pp.1725-1781, dec. 2005.

(TANENBAUM, 1995) TANENBAUM, Andrew S. Distributed operating systems. Upper Saddle River: Prentice-Hall, 1995. 614p. ISBN 0-13-219908-4.

(TANENBAUM, 1996) TANENBAUM, Andrew S. Computer networks. 3 rd ed. Upper Saddler River: Prentice-Hall PTR, c1996. 813p. ISBN 0133499456.

(VIEIRA, 1999) VIEIRA, S. Estatística experimental. 2. ed. São Paulo: Atlas, 1999. ISBN: 85-224-2113-7.

(WEI et al, 2005) WEI, Yuan et al. Performance analysis of NPB benchmark on domestic tera-scale cluster system, High Performance Computer Technical Report. Chinese Academy and Sciences Institute, 2005. Disponível em: <http://www.nas.nasa.gov/Resources/Softwa re/ware/npb.html>. Acesso em: 25 jul. 2008.

(WEISSMAN, 1999) WEISSMAN, Jon B. Fault tolerant computing on the grid: what are my options? In: IEEE INTERNATIONAL SYMPOSIUM ON HIGH PERFORMANCE DISTRIBUTED COMPUTING, 1999, Redondo Beach, CA, USA. Proceedings… Redondo Beach, 1999, 12p.

(WONG & WIJNGAART, 2003) WONG, Parkson and WIJNGAART, Rob F. Van der. NAS Parallel Benchmark I/O version 2.4. In: NAS Technical Report NAS-03-002. [S.l.: s.n.], Jan. 2003.

Anexos

Causas de Variação	GL	E(QM)
Blocos	r-1	$\sigma^2 + b\sigma^2_\gamma + ab\sigma^2_\rho$
A	a-1	$\sigma^2 + b\sigma^2_\gamma + rb\frac{\sum_j^J \alpha_j^2}{a-1}$
Resíduo a	(r-1)(a-1)	$\sigma^2 + b\sigma^2_\gamma$
B	b-1	$\sigma^2 + ra\frac{\sum_k^K \beta_k^2}{b-1}$
AB	(a-1)(b-1)	$\sigma^2 + r\frac{\sum_{j,k}^{J,K} (\alpha\beta)^2_{jk}}{(a-1)(b-1}$
Resíduo b	a(b-1)(r-1)	σ^2
Total	abr-1	

Tabela 17: Formulário para o cálculo da ANOVA

Formalismo da tabela da ANOVA, isto é, como o experimento foi constituído por efeitos fixos faz com que a esperança dos quadrados médios sigam essa tabela.

```
Analysis of Variance Table

Response: W
           Df Sum Sq Mean Sq F value   Pr(>F)
Nodo        1  40520   40520  68.462 0.003698 **
I(Nodo^2)   1   7959    7959  13.448 0.035070 *
Residuals   3   1776     592
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Tabela 18: Análise de variância (ANOVA) da classe W

Análise de variância (ANOVA) da classe W.