Universidade Federal do Rio de Janeiro
Escola Politécnica
Departamento de Eletrônica e de Computação

Gerenciador de Filas de Processamento de Dados para o Cluster do CEPEL.

Autor: _____
Daniel de Souza Dias

Orientador: _____
Antônio Cláudio Gómez de Sousa

Co-orientador: _____
André Emanoel Rabello Quadros

Examinador: _____
Aloysio de Castro Pinto Pedroza

Examinador: _____
Miguel Elias Mitre Campista

DEL

Fevereiro de 2011

UNIVERSIDADE FEDERAL DO RIO DE JANEIRO
Escola Politécnica – Departamento de Eletrônica e de Computação
Centro de Tecnologia, bloco H, sala H-217, Cidade Universitária
Rio de Janeiro – RJ CEP 21949-900

Agradecimento

Agradeço aos meus pais, irmã e toda a minha família que, com muito carinho e apoio não mediram esforços para que eu chegasse até esta etapa da minha vida.

Ao meu orientador André Quadros pela paciência na orientação e incentivo que tornaram possível a conclusão desta monografia.

Ao professor e co-orientador Antônio Cláudio por seu apoio e inspiração no amadurecimento dos meus conhecimentos e conceitos que me levaram a execução desta monografia.

Aos amigos e colegas pelo incentivo, apoio e bom humor constantes.

Aos professores e funcionário do Departamento de Engenharia Eletrônica e de Computação, por todo o auxílio acadêmico e profissional que recebi.

Resumo

A computação paralela vem assumindo um papel muito importante no mundo como um todo. Usando as tecnologias atuais, estamos chegando ao limite da frequência de operação do núcleo de um processador de dados, com isso, a nova abordagem adotada no mundo de desenvolvimento de softwares de alto desempenho é o processamento de dados de forma paralela.

No caso do CEPEL, sem a computação paralela não seria possível o estudo e a predição de dados para operação de geração de energia elétrica em tempo hábil, onde os valores estimados usando os softwares estariam disponíveis depois do tempo de aplicação desses dados.

Para que os softwares desenvolvidos utilizando o paralelismo sejam executados de forma eficiente é necessário que exista um controle dos computadores envolvidos nesse processo. É importante que cada nó de processamento esteja ocupado com somente a tarefa de processamento que foi designada a ele, usando o máximo de recursos possíveis, sem concorrência de outras tarefas.

O presente trabalho tem como objetivo desenvolver um software para o gerenciamento dos processos dos diversos computadores que formam um Cluster de processamento de dados, tomando as devidas precauções para melhorar a eficiência do processamento paralelo.

Palavras-Chave: processamento paralelo, cluster, gerenciamento de fila de casos.

Abstract

Parallel computing has assumed a very important role in the world. Using current technologies, we are reaching the limit of the frequency of a processor core. So, the new approach adopted in the world of software development is the parallel data processing.

In the case of CEPEL, without parallel computing would not be possible to study and prediction of data for operation of electric power generation in time, where the values estimated using the software would be available after the time of application of such data.

For software developed using parallelism to run efficiently there must be a control of computers involved in this process. It is important that each processing node is allocated with only the processing task that was assigned to it, using as many resources as possible, without competition from other tasks.

This study aims to develop a process management software to several computers that form a cluster of data processing, taking care to improve the efficiency of parallel processing.

Keywords: parallel processing, clustering, queue management of cases.

Siglas e Definições

API – Do inglês Application Programming Interface, que significa Interface de Programação de Aplicações é um conjunto de rotinas e padrões estabelecidos por um software para a utilização das suas funcionalidades por programas aplicativos que não querem envolver-se em detalhes da implementação do software, mas apenas usar seus serviços.

caso - É um conjunto de arquivos contendo os dados de configuração dos diferentes modelos energéticos de planejamento da expansão, geração e operação do sistema elétrico brasileiro.

CEPEL – Centro de Pesquisas de Energia Elétrica.

cluster – Do inglês aglomerado, é formado por um conjunto de computadores, que utiliza um tipo especial de sistema operacional classificado como sistema distribuído.

Encad – Sistema de Encadeamento de Modelos Energéticos.

fifo – Do inglês First In, First Out, que significa primeiro a entrar, primeiro a sair. Refere-se a estruturas de dados do tipo fila.

IDE – Do inglês Integrated Development Environment ou Ambiente Integrado de Desenvolvimento, é um software que reúne características e ferramentas de apoio ao desenvolvimento de um software com o objetivo de agilizar este processo.

Java – Linguagem de programação orientada a objeto.

Java JRE – Do inglês Java Runtime Environment. Significa Ambiente de Tempo de Execução Java.

JDK – Do inglês Java Development Kit, significa Kit de Desenvolvimento Java. Consistem em um conjunto de utilitários que permitem criar sistemas de software para a plataforma Java. É composto por compilador e bibliotecas.

LINUX – Sistema operacional baseado em UNIX.

log – Registro das transações ou atividades realizadas em um sistema de computador.

MVC – Model-view-controller é um padrão de arquitetura de software que visa a separar a lógica de negócio da lógica de apresentação, permitindo o desenvolvimento, teste e manutenção isolado de ambos.

MPI – do Inglês Message Passing Interface, é um padrão para comunicação de dados em computação paralela.

nós – Menor parte de um cluster. Representa um computador dedicado para processamento.

PBS – Do Inglês Portable Batch System é um software que executa escalonamento de tarefas baseadas em arquivos de lote.

RMI – Ou Remote Method Invocation é uma interface de programação que permite a execução de chamadas remotas em aplicações desenvolvidas em Java. É uma das abordagens da plataforma Java para prover as funcionalidades de uma plataforma de objetos distribuídos.

shell – É a interface de texto do sistema operacional.

shell script – É a linguagem usada para processamento de vários commandos ao mesmo tempo pela interface de texto do sistema operacional.

UFRJ – Universidade Federal do Rio de Janeiro

Sumário

Lista de Figuras

Capítulo 1 - Introdução

1.1 – Tema

O tema do trabalho é a produção de um software capaz de gerenciar o acesso aos recursos de processamento disponíveis em um cluster.

1.2 – Delimitação

Pretende-se criar um sistema de fila, onde o usuário conseguirá alocar recursos computacionais do cluster para processamentos diversos. O acesso aos recursos deve ser de forma transparente, sem que seja necessário o conhecimento da infraestrutura do cluster, sendo disponibilizado uma interface central de acesso. Esse sistema permite um melhor controle de acesso aos recursos, facilitando o uso pelos usuários, que não se preocupam com cada máquina em que serão executada as rotinas de processamento. A motivação deste trabalho é a produção de uma solução personalizada a ser integrada ao pacote de execução dos modelos energéticos em cluster, o software Encad, produzidos pelo Centro de Pesquisas de Energia Elétrica ou CEPEL.

1.3 – Justificativa

O CEPEL desenvolve softwares que são usados por diferentes empresas com necessidades e recursos diversificados, por isso é necessária a criação de ferramentas voltadas para as reais necessidades dos produtos que o CEPEL desenvolve. Uma dessas ferramentas é o controle do processamento de dados, usado para os estudos de otimização de operação energética. Atualmente esse controle é feito por um software open source chamado de TORQUE RESOURCE MANAGER, que só possui suporte do

fórum da comunidade, em língua estrangeira e é de complicada instalação, precisando de pessoal especializado para isso.

Com a produção de um software de gerenciamento de processos, o suporte ao software passará a ser efetuado pelo CEPEL, atendendo de forma mais rápida e precisa às necessidades dos seus clientes. Existirá a integração com o software "Sistema de Encadeamento Encad" e com a "Interface do Modelo Newave", sendo assim possível a implementação pelo CEPEL de soluções computacionais mais completas, sem dependência de gerenciadores externos.

1.4 – Objetivos

O principal objetivo do software é controlar os recursos computacionais de um cluster. Para cumprir essa tarefa, é necessário um controle dos recursos de processamento de dados em vários computadores, ou seja, o software deve ser um gerenciador de filas de processamento de dados, capaz de gerenciar a execução de jobs e monitorar recursos disponíveis no cluster.

O software deve ter uma interface onde será possível controlar a fila de recursos através de comandos. O comando de incluir um job na fila é o principal, onde será possível, através de um arquivo de configuração contendo as informações dos processos que devem ser executados, alocar os jobs em vários nós simultaneamente. Essa seleção de nós deve ser feita de modo dinâmico, de acordo com as informações de estado de cada nó disponível, onde o software gerenciador de filas será o responsável por alocar a quantidade de nós e processadores requisitados pelo usuário ou esperar que nós que já estejam alocados sejam liberados para a execução de uma nova tarefa.

A interface de gerenciamento com o usuário será feita através de linha de comando. Já a interface com o software "Interface do Newave no Encad" será feita através da linha de comandos e através da criação de uma API para o gerenciador. Esses dois métodos de interface são necessários para uma maior integração com outros produtos do CEPEL.

1.5 – Descrição

Este trabalho tem como requisito principal a produção de um software que seja capaz de gerenciar processamentos de dados. O controlador deve possuir regras para que seja possível o controle do fluxo de processamento. Para isso, foi feito o seguinte projeto, dividindo em módulos as funções.

Os recursos disponíveis no cluster devem ser alocados assim que disponíveis, mas se um job está requisitando mais recursos do que o disponível no momento, é necessário um armazenamento temporário desse job até que os recursos estejam disponíveis. Esse controle será feito através de uma fila, onde serão seguidas regras de prioridade e estado. Esse controle é feito pelo Módulo Escalonador.

O usuário então deve ser capaz de submeter casos, de retirar casos da lista de execução, de pausar casos que ainda não entraram em execução, reiniciar casos pausados. Esses são os controles necessários para a implementação do controle da fila. Para a visualização das condições da fila é necessário uma função para listar os casos que estão na fila, mostrando todas as informações requisitadas. Essas são as funções do Módulo de Interface.

Os recursos disponíveis no cluster devem ser medidos em tempo real, sendo um atraso de poucos segundos aceitável para o seu funcionamento. Sua disponibilidade será medida a cada vez que esses dados forem necessários. É necessária a instalação de um software para o monitoramento e para a execução de comandos vindo do controlador em cada um dos nós que existem no cluster. Esse software se comunicará através da rede interna com o Módulo Escalonador, sobre o protocolo TCP, utilizando o recurso de Remote Method Invocation (RMI) [3] presente na linguagem Java. Esse módulo será o Módulo Monitor de Recursos. Um melhor explicação do porquê foi escolha essa ferramenta se encontra na seção 4.2.2.

O Módulo de Interface será composto por dois módulos distintos. Existe a interface em modo texto, onde o usuário através da linha de comando do ambiente onde o software estiver instalado irá executar os comandos disponíveis. O segundo módulo será a interface com o software "Interface do Newave no Encad". Para essa função, será criada uma API simples em Java, contendo as principais funções do software controlador.

Com isso, o software será composto de 3 módulos, cada um executando uma função específica e sendo executado em processos distintos. Essa distribuição está melhor exemplificada na Figura 1.

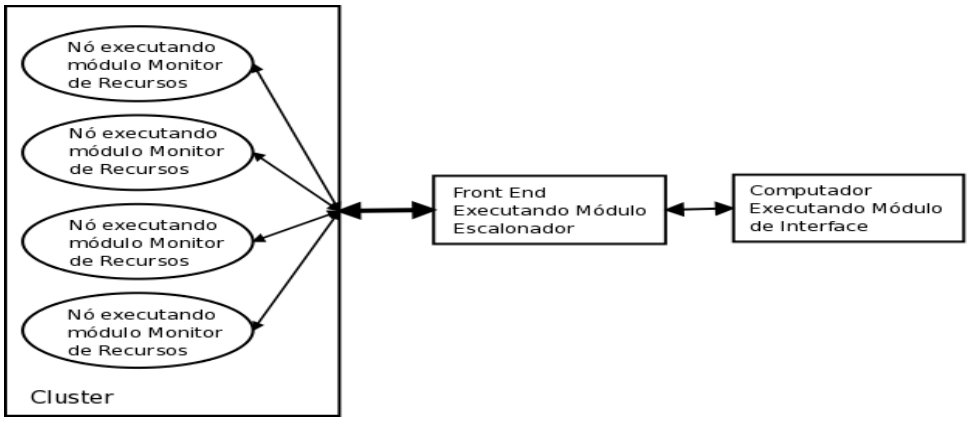

Figura 1: Distribuição do sistema.

O software ainda exige que exista uma compatibilidade entre as diversas versões de LINUX, sendo então produzido em Java[4] para que seja minimizado o risco de incompatibilidade entre as diversas versões.

Além desse capítulo de introdução, este trabalho está organizado em mais 5 capítulos.

No capítulo 2, **Motivação**, será apresentado o ambiente onde o software está inserido nas atividades do CEPEL e como o software será utilizado para auxiliar no desenvolvimento das atividades do mesmo.

O capítulo 3, **Descrição do Software e metodologia**, apresenta de forma resumida a proposta das funções do software e a sua divisão em módulo. Serão apresentadas também as metodologias que foram usadas durante o projeto e as ferramentas que foram utilizadas.

O capítulo 4, **Requisitos do sistema**, serão apresentadas as metodologias aplicadas para a construção do documento de especificação de requisitos. Este documento define os requisitos funcionais, os requisitos de dados e as restrições que compõem o sistema. A finalidade desse documento é o estabelecimento de um escopo bem definido para a implementação do software, servindo de base para o processo de desenvolvimento.

Baseado no documento de especificação de requisitos, o projeto do sistema é apresentado no capítulo 5, **Projeto do Sistema**. Nesse capítulo será explicitada a arquitetura do sistema, descrevendo as soluções apresentadas para a definição e geração. Serão discutidas as decisões de projeto, os casos de uso gerados, os diagramas de estado e de classes. Nesse capítulo também serão detalhadas as tecnologias utilizadas para a implementação da solução proposta.

No capítulo 6, **Resultados**, são exibidos os resultados obtidos utilizando o aplicativo para o gerenciamento de execução dos modelos utilizados pelo CEPEL.

No capítulo 7, **Conclusão**, serão descritas as conclusões obtidas a partir do trabalho executado, ressaltando suas contribuições e as perspectivas de trabalhos futuros.

No anexo A, **Especificação de requisitos de software**, é apresentado o documento de especificação de requisitos do sistema.

No anexo B, **Projeto**, é apresentado o documento de projeto do sistema.

Capítulo 2 - Motivação

2.1 - Introdução

Nesse capítulo serão expostos os principais fatores que levaram a escolha desse tema como projeto de final de curso. Primeiro é descrito onde o software está localizado, onde que ele será utilizado e em conjunto com que softwares.

Depois se chegará a conclusão do porquê esse software está sendo desenvolvido, tentando no caminho mostrar a importância que o projeto no qual o software está contido tem para o setor energético brasileiro.

2.2 - Cepel

O Centro de Pesquisas de Energia Elétrica ou Cepel, foi criado em 1974, sendo a sua sede localizada no Rio de Janeiro, na Ilha do Fundão. Ele faz parte do grupo Eletrobrás, tendo mais de 30 anos de experiência em pesquisa e desenvolvimento relacionado a geração, transmissão e distribuição de energia elétrica. É considerado o maior centro de pesquisas em energia elétrica do Hemisfério Sul.

2.3 – Encad

O objetivo principal do programa Encad é facilitar a utilização da cadeia de modelos propostos pelo CEPEL para o planejamento da expansão e operação energética através de interfaces gráficas e realizar o encadeamento destes modelos, ou seja, permitir que o conjunto de informações produzidas por modelos de nível hierárquico

superior sejam compartilhadas com os modelos de nível hierárquico inferior, realizando validações na alteração de um caso inserido em uma árvore hierárquica e ainda tarefas automáticas de sincronismo entre casos encadeados. Sua arquitetura possibilita a diminuição e até mesmo a eliminação de diversas dificuldades associadas ao trabalho com os arquivos de dados dos modelos, tornando a configuração, o fluxo dos dados e a execução dos diversos modelos tarefas simples.

Com o Encad é possível estruturar o trabalho, permitindo que diversos casos do mesmo modelo e/ou de outros modelos sejam agrupados em estudos conforme necessidade do usuário. Após a execução de um modelo, o Encad disponibiliza a visualização de resultados através de gráficos e/ou tabelas de valores com possibilidade de filtrar a enorme gama de resultados que cada modelo provê para informações específicas para o estudo do usuário.

Atualmente o Encad possui interfaces gráficas para os seguintes modelos da cadeia: Sinv, Melp, Newave, Suishi, Gevazp, Confint, Previvaz, PrevivazH, Decomp, Dessem e Newave com Processamento Distribuído.

2.4 – Newave

O Software NEWAVE é um modelo desenvolvido pelo CEPEL, que tem por objetivo a otimização da operação energética de médio/longo prazo do sistema interligado nacional (SIN). Integrante da cadeia de modelos para otimização energética, também desenvolvida pelo CEPEL, o NEWAVE é utilizado por vários atores do setor energético brasileiro. Podemos citar, entre outros, o Operador Nacional do Sistema Elétrico (ONS) que usa esse modelo no planejamento da operação energética de médio prazo, através dos planos mensais de operação (PMO) e o Ministério de Minas e Energia (MME) que usa esse modelo nos estudos de planejamento da expansão da geração de curto-prazo (Plano Decenal de Energia);

Dentre os principais resultados fornecidos pelo NEWAVE, destacam-se a função de custo futuro, que representa o custo de operação do estágio atual até o final do horizonte de planejamento e índices de desempenho do sistema, tais como o risco

anual de déficit por falta de energia, o valor esperado do custo de operação, o valor esperado do custo marginal de operação e o valor esperado da energia não suprida.

Este Software emprega técnicas de otimização estocástica, trabalha com horizontes de planejamento de 5 a 30 anos, discretizados em etapas mensais, e modela o processo estocástico de afluências às usinas hidroelétricas através de diversos cenários de afluências de comprimento igual ao horizonte de planejamento, o que pode demandar algumas horas de execução. Esses modelos, ao serem paralelizados, tem o seu tempo de execução reduzido drasticamente. Podemos dar como exemplo uma execução convencional de um caso PMO. Ao ser executado convencionalmente, de forma sequencial, é gasto aproximadamente 48 horas para a conclusão dos cálculos. Ao ser executado de forma paralela, com 32 núcleos fazendo os cálculos, o tempo de conclusão é de aproximadamente 3,2 horas.

2.5 – Labcin

Labcin, ou Laboratório de Computação Intensiva, foi concebido para prover ao CEPEL um ambiente para desenvolvimento e execução de aplicações de alto desempenho. É usado para viabilizar diversos estudos, como para o Ministério de Minas e Energia e para o Planejamento da Expansão e da Operação. Além desses estudos, serve de plataforma de aprimoramento na modelagem matemática dos programas NEWAVE e DECOMP.

Foi inaugurado em 2006 e está localizado na Sede do CEPEL, na Cidade Universitária. Atualmente é coordenado pelo pesquisador Roberto José Pinto.

2.6 – Cluster

Atualmente o Labcin possui 3 clusters de alto desempenho distintos, o primeiro é composto por 32 núcleos de processamento Intel Xeon de 3,2 GHz cada, com um total de 64 Gb de memória RAM. O segundo e maior cluster é formado por 336 núcleos de processamento Intel Xeon de 2,6 GHz cada, com um total de 672 Gb de memória RAM. O último cluster é formado por 56 núcleos de processamento Intel Xeon de 3,0 GHz cada, com um total de 56 Gb de memória RAM.

2.7 – MPI

Em 1992 foi criado um fórum para discutir uma padronização para trocas de mensagens. Este fórum foi composto de pesquisadores, programadores, usuários e fabricantes de computadores que definiram a sintaxe, a semântica e o conjunto de rotinas para troca de mensagens em ambientes com memória distribuída. Esse conjunto de rotinas foi agrupado numa biblioteca chamada de MPI (Message Passing Interface). Estas rotinas, ou funções, são implementadas ao longo do código do programa para viabilizar a distribuição de tarefas entre os processadores participantes do ambiente distribuído. Então, o objetivo principal do MPI é prover um amplo padrão para escrever códigos com passagem de mensagens de forma prática, portátil, eficiente e flexível.

2.8 - Torque

O *TORQUE Resource Manager* é um gerenciador de recursos de código aberto que fornece controle sobre tarefas em lote e nós de computação distribuída. Atualmente é desenvolvido pela comunidade aberta, com base no projeto original *PBS*, que não é mais desenvolvido.

Este sistema tem por finalidade alocar recursos computacionais no cluster para o processamento dos modelos de forma eficiente. Isso permite uma abstração de que todo recurso esta disponível em um único ponto de entrada, isto é, o usuário não terá a necessidade de entrar num determinado nó do cluster para poder usar o seu recurso computacional, ele apenas submete a tarefa que o sistema de gerenciamento aloca os recursos necessários para essa tarefa.

O *TORQUE* é atualmente a escolha do CEPEL para controlar o ambiente de processamento paralelo do Labcin.

2.9 - Conclusão

O Cepel possuía uma demanda por um software capaz de gerenciar uma fila de processos num cluster, que conseguisse se integrar de forma mais fechada ao modelo que eles desenvolvem para o planejamento de expansão e operação energética, o Encad. O software que é usado atualmente para o controle de processos no Cluster do Labcin é o Torque, que apesar de ser um software de bons recursos não é facilmente integrado ao pacote do Encad.

Foi então proposta a criação de um software em uma linguagem multiplataforma, capaz de realizar tarefa semelhante ao Torque, só que integrada ao software Encad, de forma que fosse possível a customização das suas funções de acordo com as necessidades do Cepel.

Capítulo 3 - Descrição do Software e metodologia

3.1 - Descrição do software

O software tem como principal função o gerenciamento da execução de modelos computacionais em vários nós processadores de um mesmo cluster.

Para executar essa tarefa, é necessário que exista uma Fila, onde esses modelos computacionais, também chamados de casos, entrem em ordem de chegada, ou seja, o primeiro caso a entrar seja também será o primeiro a sair. Esse tipo de fila é denominada First In First Out, ou FIFO, e com essa fila é possível gerenciar a ordem de execução dos casos. Essa fila foi escolhida por ser a utilizada no Cepel, tendo ao longo do tempo demonstrado ser uma boa escolha.

Não basta a organização dos casos, também é necessário a comunicação com os diversos nós de processamento que estão a disposição para os cálculos. Para isso, deve ser criado um componente com o intuito de monitorar o estado de cada um dos nós que compõem o sistema, recebendo e enviando informações ao módulo responsável pelo gerenciamento da fila. É necessário também a divisão dos computadores que irão processar os dados em unidades homogêneas, ou seja, separar as máquinas semelhantes em listas contendo somente um tipo de computador. Isso é desejável para um melhor funcionamento do protocolo MPI e do modelo Newave, que necessitam de uma plataforma homogênea para ter o seu melhor desempenho.

Por último, é necessária uma interface por onde o usuário possa controlar e verificar o estado em que se encontra o gerenciador da fila de casos e da lista de nós disponíveis, para que os processos a serem executados ocorram da melhor maneira possível.

Para ilustrar a separação do software, foi criado um diagrama mostrado na Figura 2, onde temos os três diferentes módulos interagindo. Sua interação pode ser

melhor visualizada na seção 3 do documento de Projeto de Software, que se encontra no apêndice B.

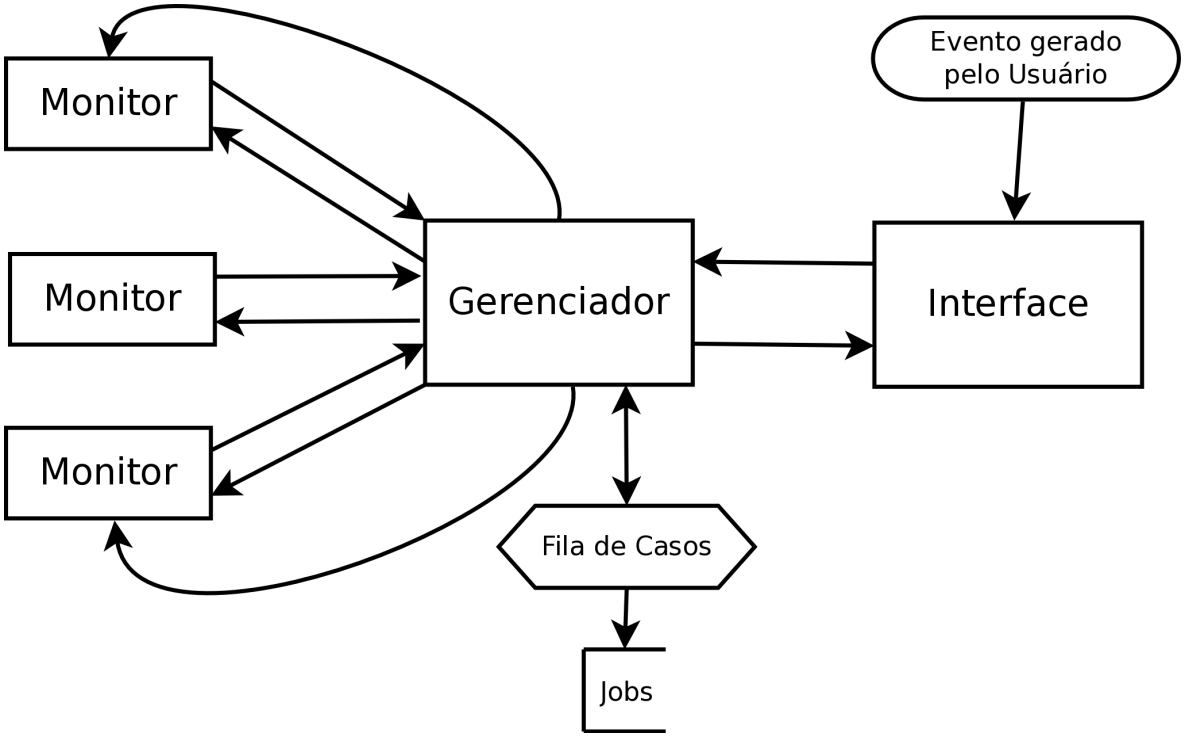

Figura 2: Diagrama dos componentes básicos do software.

3.2 – Ferramentas utilizadas

A linguagem escolhida para a implementação do projeto foi Java, principalmente devido a sua característica multiplataforma e a facilidade de comunicação entre as partes separadas do software.

Foi escolhida a linguagem na versão 6, versão mais atual disponível na época da implementação, em que todas as ferramentas utilizadas possuíam total compatibilidades. É bom ressaltar que foi utilizada o protocolo Remote Method Invocation ou RMI para a comunicação entre as diferentes partes do software.

Como ambiente de desenvolvimento foi escolhida a IDE Netbeans 6.9.1, a versão mais atual disponível na época do desenvolvimento, que conta com o kit de

desenvolvedor JDK 1.6, que contém todas as ferramentas necessárias para a compilação e execução de códigos Java.

Para os testes, foi utilizado o software Virtual Box para a criação de um ambiente onde fosse possível a emulação de um cluster virtual. Foram criadas 4 máquinas virtuais rodando Ubuntu 10.10, para que fosse possível a simulação da comunicação entre as diversas máquinas.

Para a criação dos documentos, foi utilizado o editor de texto Microsoft Word. Para a criação dos diagramas foi utilizado o software DIA.

3.3 – Metodologias utilizadas

Para o desenvolvimento desse sistema, foi adotado o modelo em cascata, em que os estágios são apresentados em sequência, como uma cascata, seguindo as etapas de especificação de requisitos, análise, projeto e implementação. Dessa forma, só quando todos os requisitos forem enunciados, tiverem sua completeza e consistência analisadas, e tiverem sido documentados em um documento de especificação de requisitos, é que se pode realizar as atividades de análise e projeto do sistema.

Foi utilizado o método de orientação a objetos para o desenvolvimento do software, tendo como intuito melhorar o processo de modelagem do mundo real no domínio do problema em um conjunto de componentes de software que seja o mais fiel na sua representação deste domínio.

Para a comunicação entre os diversos módulos foi escolhida a arquitetura de objetos distribuídos, onde a distinção entre cliente e servidor é removida. Nessa arquitetura, os componentes fundamentais do sistema são objetos que fornecem uma interface para um conjunto de serviços fornecidos. Outros objetos podem chamar esses serviços sem distinção lógica entre um cliente e um servidor. Esses objetos podem ser distribuídos através de uma rede ou concentrados em um mesmo computador.

Capítulo 4 - Requisitos do Sistema

4.1 - Introdução

Nesse capítulo discutiremos o processo de especificação de requisitos e a construção do documento de requisitos da ferramenta proposta. No item 4.2 serão abordadas as restrições do projeto, apresentando tecnologias utilizadas para o desenvolvimento do sistema e justificando suas escolhas. No item 4.3 e no item 4.4 serão discutidos os requisitos do software e o processo de identificação e especificação dos requisitos.

O objetivo desse capítulo é apresentar o documento de especificação de requisitos produzido, discutindo a metodologia utilizada para a definição e construção do documento. A versão final do documento de requisitos de software produzida para esse projeto se encontra no anexo A.

4.2 – Restrições do projeto

Esse projeto foi idealizado tendo um foco bem definido. Ele deve fazer o controle dos casos e ter uma interface com o usuário, uma outra interface com um software Encad já em produção e uma terceira interface com o sistema operacional dos nós de processamento do Cluster. Algumas restrições adicionais foram impostas pelo Cepel, como a linguagem que deveria ser utilizada e o método de comunicação entre as diversas parte. Isso aumentou as restrições impostas ao software, já definindo previamente algumas tecnologias que deveriam ser utilizadas.

Com base nas restrições descritas no parágrafo anterior e no documento de especificação de requisitos que se encontra no anexo A, a seção a seguir apresenta as técnicas a serem utilizadas no desenvolvimento da solução, tendo junto uma descrição de cada técnica, justificando a sua escolha.

4.2.1 Linguagem Java

Java é uma linguagem de programação orientada a objetos muito utilizada atualmente para a implementação de projetos por ser uma linguagem orientada a objetos, com uma grande portabilidades, já que é executada em máquinas virtuais. Além disso, possui amplas bibliotecas, facilitando a implementação de rotinas.

Considerando as características desse projeto, que precisa de portabilidade e facilidade de instalação e a preferência do Cepel pela linguagem Java, ela foi a escolhida.

4.2.2 Remote Method Invocation

Remote Method Invocation, ou RMI, é uma interface de programação que permite a execução de chamadas remotas em aplicações desenvolvidas em Java. É utilizada pela linguagem Java para prover as funcionalidades de objetos distribuídos. Através da utilização da arquitetura RMI é possível que um objeto ativo em uma máquina virtual Java possa interagir com objetos de outras máquinas virtuais Java, independente da localização desses objetos.

Foi escolhida essa API pela a facilidade de implementação de interfaces de comunicação, diminuindo expressivamente a dificuldade de implementar o código, e conseqüentemente levando a um menor número de erros. Além disso, a API do RMI tem políticas de segurança, que ajudam a aumentar a confiabilidade do sistema.

4.2.3 JDK

JDK ou Java Development Kit é um conjunto de utilitários que permitem criar sistemas de software para a plataforma Java. Ele é composto de compiladores e de bibliotecas.

Nesse projeto foi utilizada a versão JDK 6 update 23, por ser a última disponível na época.

4.2.4 IDE NetBeans

IDE, do inglês Integrated Development Environment, significa ambiente de desenvolvimento integrado. Consistem um uma plataforma com as ferramentas necessárias para o desenvolvimento de aplicações. No caso desse projeto, foi utilizada a plataforma Netbeans, que tem vastos recursos para a programação em Java e é multiplataforma, sendo executado em Windows, Linux e MacOS, fator determinante para a escolha já que o desenvolvimento ocorreu nas três plataformas simultaneamente.

4.2.5 Virtual Box

É um software de virtualização desenvolvido pela Oracle que visa criar ambientes para a instalação de sistemas distintos. Sua função é implementar através de um software uma maquina, que executa softwares como um computador real e independente. Ele permite a instalação e utilização de um sistema operacional dentro de outro, totalmente isolados, mas compartilhando o mesmo hardware.

O motivo do uso de uma ferramenta de virtualização foi o de conseguir fazer testes em plataformas distintas, não se apegando a somente uma plataforma. Os testes serão feitos em máquinas virtuais com diferentes versão de sistemas operacionais, como por exemplo o CENTOS 5.4 e o Ubuntu 10.10. A instalação e configuração de diversos sistemas se torna mais fácil quando é feita nesse ambiente virtual, ganhando tempo e praticidade, já existe um o menor tamanho físico do sistema todo e a possibilidade de cópia de maquinas inteiras de modo rápido.

Outro fator determinante para o uso de virtualização nos testes foi a indisponibilidade de um teste no Cluster do Labcin, que precisa estar disponível 24h por dia, não sendo possível a alteração de sua configuração para testes sem que ele fique algum tempo fora de operação.

4.3 – Requisitos de software

Quando a ideia do software foi proposta, ela possuía somente uma descrição em alto nível. Era sabido somente a função principal que o software deveria executar e algumas poucas restrições. Para fazer a definição das funcionalidades, foi proposta a criação de um documento de Requisitos de Software, com o intuito de formalizar as funcionalidades e características do sistema.

Os requisitos podem ser características ou ações que o sistema deve realizar. O documento deve ser capaz de descrever suas características de forma clara, de modo que tanto o cliente como o desenvolvedor possam ter a mesma compreensão do funcionamento do sistema proposto. O documento de especificação de requisitos de software tem como objetivo descrever "o que" o sistema deve fazer, não se preocupando com o "como fazer", que deverá ser abordado em outro documento antes do início da implementação.

Os principais requisitos desse sistema que foi proposto é a capacidade de organizar o processamento de uma fila de casos, de forma que não ocorra o processamento de mais de um trabalho de cálculo em um nó do cluster ao mesmo tempo. O sistema também deve ter a possibilidade de se comunicar e observar o estado de cada nó que compõe o sistema, executando as tarefas que foram direcionadas e esse nó. Além disso, a ferramenta deve ser capaz de interagir com outros sistemas e com os usuários, para que as tarefas sejam executadas de forma autônoma.

Esses requisitos estão descritos mais detalhadamente no documento de especificação de requisitos, que se encontra no apêndice A, sendo importante a sua leitura para uma melhor compreensão dos requisitos do sistema.

4.4 – Identificação e especificação dos requisitos

O processo de identificação de requisitos envolve uma metodologia que visa a correta identificação e especificação das funcionalidades do sistema. Devemos buscar

atender a todas as necessidades dos usuários, verificando com o próprio se conseguimos alcançar a correta visualização do problema. Cada requisito lida com objetos ou classes, os estados que podem possuir e as funções que serão realizadas para modificar os estados ou as características do objeto.

Com o objetivo de entender o problema a ser solucionado, foram utilizadas algumas técnicas para determinar o que era necessário para o correto funcionamento do sistema. Em primeiro lugar, foi estudada e infra-estrutura de onde o programa seria instalado primariamente. As características do laboratório do Labcin do Cepel foram levadas em conta como base para a instalação do sistema. Além desse estudo, foi discutido com a equipe de desenvolvimento do Encad para saber quais eram os padrões das instalações de parceiros do Cepel. Em seguida, foram listadas as principais características e funcionalidades, com um nível alto de abstração. Essa parte foi levada em conta na hora de definir as restrições do projeto principalmente.

Após esse levantamento inicial, foi observado o ambiente existente de gerenciamento de processamento de casos. Nessa etapa, foi estudado o funcionamento do software PBS, do software Torque e da biblioteca MPI. A biblioteca MPI é usada pelo modelo Newave para o paralelismo das operações, e ela necessita de alguns dados que são passados pelo software Torque, responsável pelo gerenciamento da fila de casos. O software PBS foi estudado por ter sido um dos primeiros softwares a implementar a funcionalidade de gerenciamento de filas de processamento para grandes clusters, sendo substituído posteriormente pelo software Torque. Nessa etapa, foram lidos os documentos de guia de usuário da biblioteca MPI, guia de administrador do software PBS, guia de usuário do software PBS, guia de administrados do software Torque. Esses documentos se encontram referenciados na bibliografia.

De posse dessas informações e da bibliografia indicada, foi possível descrever os requisitos do software no documento de especificação de requisitos de Software. Com o intuito de assegurar a correta especificação dos requisitos, e que todos os requisitos haviam sido corretamente levantados, o documento passou por algumas etapas de verificação. Essas etapas de verificação e validação procuravam garantir que a descrição feita no documento refletia o que foi proposto inicialmente.

O documento de especificação de requisitos foi construído baseado nas normas apresentadas na disciplina de Engenharia de Software do departamento de Engenharia Eletrônica e de Computação, ministrada pelo professor Antônio Cláudio. Será feita uma

breve descrição de cada tópico que existe no documento de especificação de requisitos para que se tenha um melhor entendimento da sua função.

O documento é iniciado fazendo uma breve descrição de sua finalidade, com as definições e as referências usadas para a sua confecção. Depois, no segundo capítulo é feita uma descrição geral, onde as perspectivas do produto e suas funções são descritas de forma geral. Nessa parte também são levantadas as características dos usuários e as restrições gerais.

Depois são apresentadas as interfaces externas do software. Foi idealizado que o software teria duas interfaces, uma de linha de comando e uma com o software Encad. Foi também levantado que o software dependerá de conexões de redes baseadas em TCP/IP para se comunicar internamente.

Nos requisitos funcionais, foi feito um diagrama de casos de uso, com todas as funcionalidades que o software deve possuir. Cada uma das interações foi descrita detalhadamente, e se encontram no anexo B para a leitura. Houve a necessidade de explicar com maior clareza a função do escalonador de tarefas, então foi criado um diagrama de atividades, com o intuito de facilitar a compreensão do funcionamento da principal lógica de funcionamento do software.

Para mostrar os vários estados que um caso pode adquirir no funcionamento do software, foi criado um diagrama de estados, detalhando cada estado que um caso pode ter desde que ele foi incluído no escalonador.

Para completar o documento, foram levantados os requisitos de desempenho, onde foi levado em consideração que o software não precisa de respostas instantâneas, já que é idealizado para controlar processos que podem demorar horas para ser finalizados, sendo então alguns segundos pouco relevante para o correto funcionamento do software.

Capítulo 5 - Projeto do Sistema

5.1 - Introdução

Neste capítulo serão descritas as fases de produção do documento de projeto de software e a implementação do sistema em si. Serão apresentados a definição do problema, a modelagem dos dados, as decisões mais importantes que foram tomadas no processo de implementação e a solução adotada para a resolução do problema.

5.2 – Definição do problema

Durante a execução do sistema, o software deve ser capaz de gerar um arquivo Shell Script com instruções dos procedimentos que deverão ser seguidos para executar o caso. A seguir, esse arquivo deve ser distribuído para uma quantidade de nós selecionada pelo usuário no momento da inclusão. Nesse Script gerado estarão as comandos que foram submetidos pelo usuário para executar determinadas tarefas acrescido de mais algumas para ajudar no monitoramento do estado do processamento.

Uma característica importante é que o sistema deve ser resistente a falhas de comunicação. Se por algum motivo não for possível a comunicação com um dos nó, o sistema deve ser capaz de se recuperar do erro, abortando o caso que se encontra processando naquele momento.

Outra característica, usada para prevenir que o sistema fique travado, sem condições de continuar o processamento, é o impedimento da submissão de casos que requisitem um número maior de processadores do que o disponível no escalonador. Nesse caso, o sistema deve retirar o caso da fila, gerando um erro que deve ser registrado.

O acesso simultâneo que é originado por ser possível o acesso simultâneo ao software, por uma ou mais interfaces de usuário foi umas das características que teve

que ser levada em conta na hora da construção das classes. Determinadas ações não podem executar juntamente com outras. Para resolver esse problema, uma ferramenta Java foi usada. Todas as classes que tinham acesso restringido foram criadas como sendo synchronized. Essa ferramenta impede que outro acesso seja feito ao software enquanto a classe está sendo processada, ou seja, a classe é processada como um todo antes que qualquer outra instrução seja executada.

A última característica mostrada aqui que foi levada em conta é a alteração em tempo real da configurações, onde alguns valores podem ser mudados durante o tempo de execução do software. Isso é importante principalmente na parte dos nós, onde é possível adicionar ou remover nós enquanto o software estiver sendo executado.

Baseando-se nesses características do sistema e no documento de especificação de requisitos, a seguir serão apresentados a modelagem de dados idealizada, e a solução proposta para a construção do sistema.

5.3 – Modelagem dos dados

Como já foi indicado anteriormente, o sistema foi dividido em três módulos: o módulo escalonador, o módulo de interface com os nós e o módulo de interface com o usuário. A seguir, na Figura 3 é apresentado o diagrama de classes da interface do escalonador, com as explicações do que representa cada classe.

Esse módulo como um todo é responsável pela organização e armazenagem das informações dos casos. Ele também e responsável por conter as informações de configuração e as informações sobre os nós com que esse módulo do software precisa se comunicar. Todas as listas foram armazenadas como ArrayDeque, uma ferramenta da linguagem Java. Essa ferramenta é uma lista com diversas opções de acesso, facilitando o trabalho do desenvolvedor.

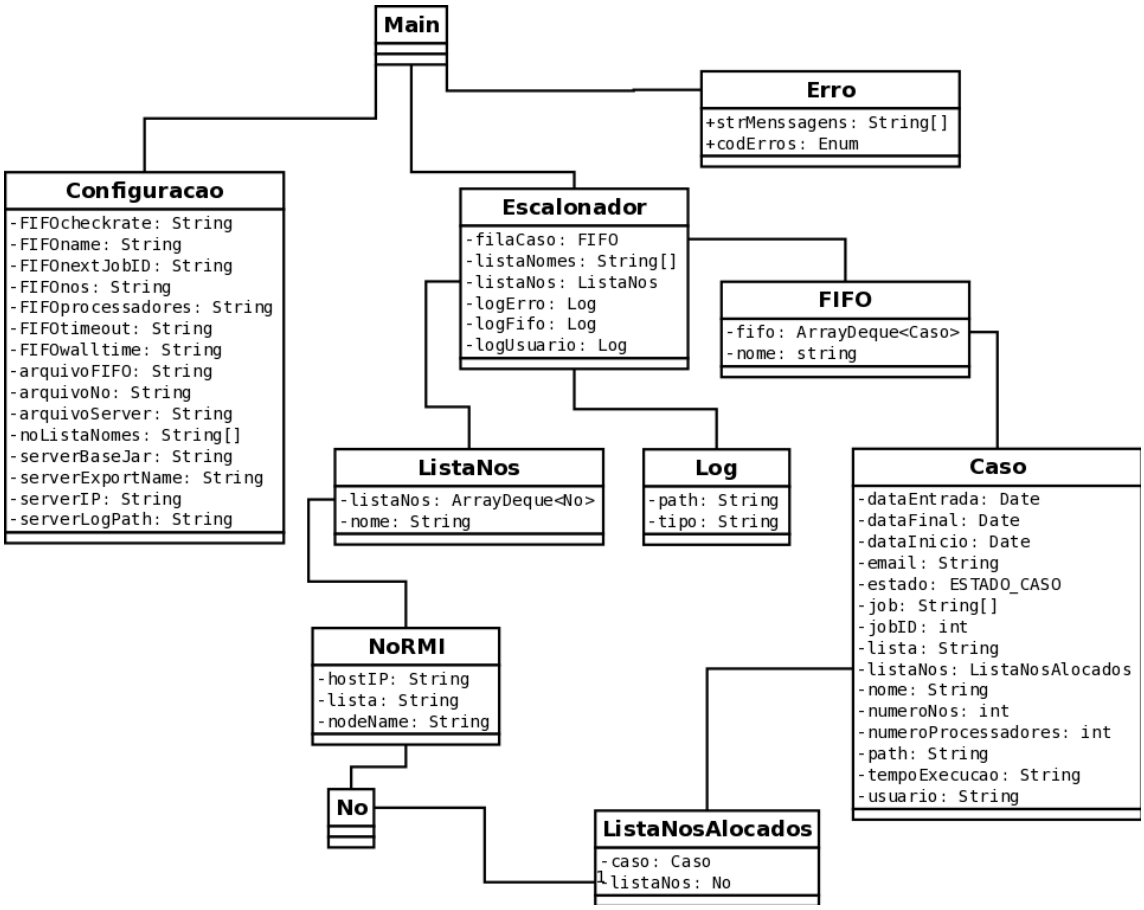

Figura 3: Classes do Escalonador.

Nesse diagrama temos como os dados devem estar organizados dentro do módulo. Abaixo, é mostrado brevemente como é feita essa separação. Para maiores detalhes, é necessário ler o Apêndice B, projeto de Software.

- Configuração: Contém os valores carregados do arquivo de configuração ou os padrões definidos pelo software;
- Erro: Contém os números de devolução de cada erro e a sua correspondente linha de texto de erro;
- Escalonador: Contém as filas de caso e as filas de nós. Contém também os locais para escrever nos arquivos logs. Nele estão contidos:
    - filaCasos: Local em que é iniciada e armazenada a fila de casos;
    - listaNomes: Lista contendo os nomes de todas as listas disponíveis;
    - listaNos: Local em que é iniciada e armazenada a lista de nós;
- Log: Contém os dados necessários para inicializar e manter em memória um arquivo de log;

- FIFO: Contém as listas de casos. Contém a seguinte fila:
  - fifo: Fila que contém a fila de casos, do tipo FIFO;
- Caso: Contém os dados de um caso submetido. Fazem parte dessa classe as seguintes variáveis:
  - dataEntrada: Data em que o caso foi submetido;
  - dataFinal: Data limite para que o caso seja finalizado;
  - dataInicio: Data em que o caso começou a ser executado;
  - email: email da pessoa que submeteu o caso;
  - estado: Estado em que se encontra o Caso. Existem cinco valores distintos para essa variável, que são: Processando, Ativo, Suspenso, Concluído e Erro;
  - job: Arquivo que foi submetido em forma de vetores de linhas;
  - jobID: Número de identificação atribuído a esse caso;
  - lista: Lista a que o caso pertence. Caso não seja definido, será escolhido o valor padrão default;
  - listaNos: Lista de nós alocados para esse caso;
  - nome: Nome com que o caso foi submetido;
  - numeroNos: Numero de nós utilizados para processar o caso;
  - numeroProcessadores: Numero de processadores utilizados para processar o caso;
  - path: Caminho em que foi chamado o Caso;
  - tempoExecucao: Tempo máximo que um caso pode ficar sendo executado;
  - usuário: nome do usuário que submeteu o caso;
- Lista Nós: Lista que contém as informações para se conectar aos nós remotos;
- NoRMI: Contém as informações para se conectar a um nós remoto;
- No: Contém as informações do nó e do caso que está executando. Esses dados são acessados através de funções;
- ListaNosAlocados: Contém a lista de nós que foram alocados a um determinado caso.

O módulo da interface de comunicação com os nós é apresentado na Figura 4. Ele é responsável por manter as informações de cada um dos nós. As informações relevantes

de serem apresentadas nesse documento são: o estado no Nó, que guarda as informações se um nó está ativo, inativo ou processando um caso; as informações de comunicação, como IP e nome do nó; e o caso que foi submetido, com as suas informações.

Para uma melhor compreensão do papel desse módulo no software, deve ser lido o documento de projeto do software, localizado no Apêndice B.

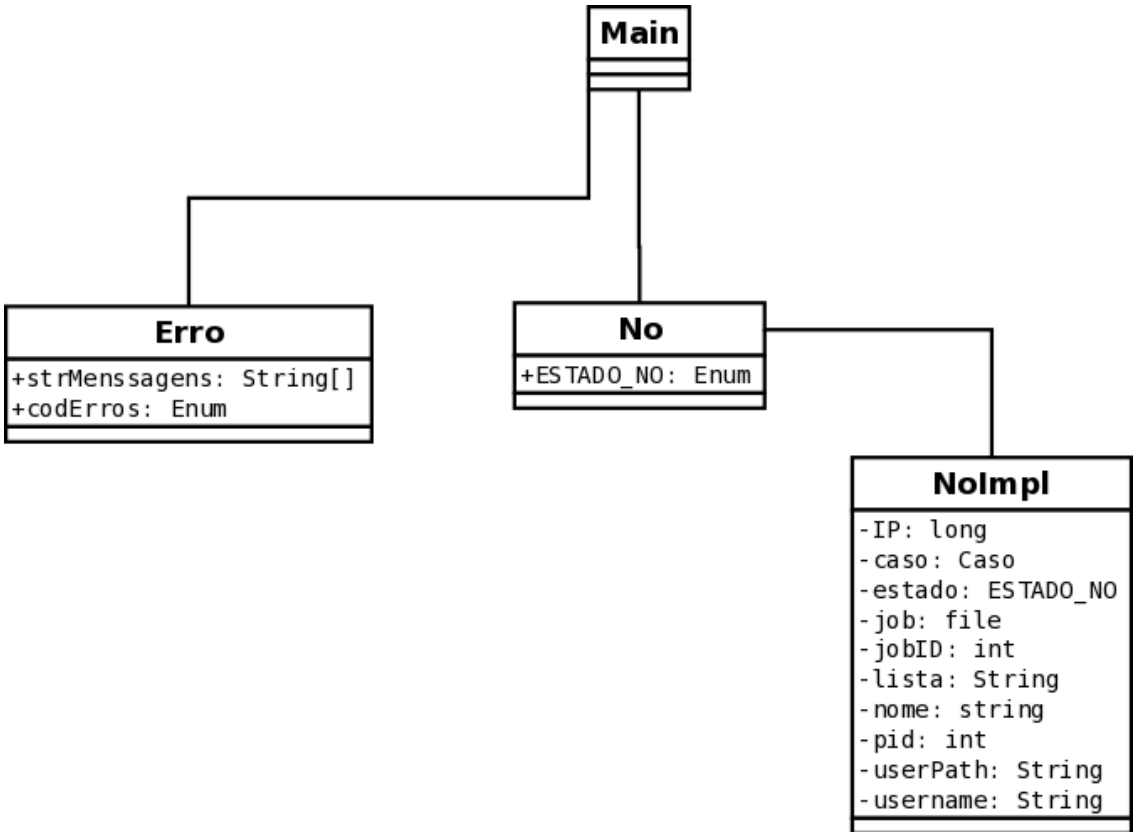

Figura 4: Classes da Interface de comunicação dos Nós

O módulo de interface com o usuário é responsável pela comunicação com o usuário. A partir desse módulo o usuário envia as informações desejadas ao módulo escalonador e consegue ler o estado em que o escalonador está naquele momento.

Esse módulo tem a implementação simples, sendo desnecessária a explicação extensa. Para uma melhor compreensão do papel desse módulo no software, é recomendável a leitura do documento de projeto de software, localizado no Apêndice B.

5.4 – Decisões tomadas durante a implementação

No começo da implementação, foi decidido que o uso da ferramenta de JavaDocs seria utilizada, para que fosse mais fácil a criação das documentações e para que uma documentação que ajudasse na hora da produção de uma API fosse produzida. Esses documentos estão disponíveis em forma eletrônica, anexados ao projeto para a facilitação do seu uso.

Depois da definição das classes, como primeiro passo para a implementação, foram criadas as suas funções. A seguir, é apresentado o diagrama de classes do módulo escalonador. Esse módulo contém as funções que são utilizadas principalmente para a comunicação com os outros módulos e para a organização dos casos no software.

As funções mais importantes serão descritas logo após a Figura 5. Para uma completa descrição, é recomendada a leitura do JavaDocs do Módulo Escalonador, que se encontra na mídia anexa ao projeto.

Figura 5: Diagrama de classes do módulo escalonador.

Funções da classe Configuração: Essa classe é responsável por ler e salvar as configurações em um arquivo texto. Abaixo estão descritas as funções mais importantes. Outras funções não descritas, que começam com a palavra get são responsáveis por devolver o valor da variável pedida. As funções iniciadas em set são responsáveis por armazenar o valor das variáveis:

carregar(): int – responsável por carregar os dados do arquivo de configuração

salvar(): int – Responsável por salvar os dados no arquivo de configuração. É chamada cada vez que um dado é modificado.

Funções da classe Interface: Essa classe é responsável por conter a interface das funções que devem ser acessadas remotamente através de RMI. A sua implementação não será feita nessa classe e sim na InterfaceImpl. As funções serão explicadas na InterfaceImpl.

Funções da classe InterfaceImpl: Essa classe é responsável por implementar a classe Interface. Ela é responsável pelo acesso a este módulo pelo módulo de interface com o usuário. As funções desse módulo são simples, só são responsáveis pelo acesso. Os nomes das funções são autoexplicativos.

Funções da classe Escalonador: Essa classe é responsável por gerenciar a fila de casos e a lista de nós. Nela estão as funções que verificam o que deve ser feito em seguida no software. Abaixo estão descritas as funções mais importantes:

criarCaso(usuário: String, path: String, job: String[]): int – Executa os procedimentos necessários para criar um caso. Esse caso deve ser adicionado a FIFO.

verificarConcluídos(): int – Verifica os casos cujo processo terminou em todos os nós. Verifica também o tempo limite de execução.

verificarInicioCaso(): int – Verifica a possibilidade de início de casos e os inicia.

verificarNos(): int – Verifica a conectividade com os nós.

verificarFIFO(): int – Verifica todas as acões que podem ser tomadas. Em seu conteúdo, ela chama as funções de verificação acima descritas.

Funções da classe ListaNos: Essa classe é responsável por armazenar a lista de nós disponíveis no sistema. Abaixo estão descritas as funções mais importantes:

alocarNos(quantidade: int, listaNosCaso: int, lista: Sring): int – Responsável por executar os procedimentos de reservar os nós para que sejam usados por um único caso.

contarNosDisponiveis(lista: String): int – Responsável por contar quantos nós estão disponíveis em determinada lista.

contarTotalNos(): int – Responsável por contar quantos nós existem em determinada lista.

verificarEstado(): int – Verifica se um nó terminou o processo mas não teve o seu estado alterado.

Funções da classe NoRMI: Essa classe é responsável por conectar aos objetos remotos que contém os dados de cada um dos nós. Abaixo estão descritas as funções mais importantes:

Conectar(): No – Essa função é responsável por conectar ao objeto remoto e prover o acesso a ele. É retornado o objeto remoto, com o qual é possível executar todas as operações como se fosse localmente.

Funções da classe No: Essa classe é responsável por armazenar os dados dos nós. Como é um objeto remoto, nessa classe temos somente as interfaces para a conexão. As funções são simplesmente de acesso aos dados remoto, por isso não serão descritas aqui.

Funções da classe ListaNosAlocados: Essa classe é responsável por fazer a conexão entre um caso e os nós determinados para executar esse caso. Nela está contido o caso a ser executado e uma lista com os nós que foram alocados para executar as tarefas desse caso. Abaixo as principais funções serão descritas:

atualizarEstadoNos(estado: ESTADO_NO): int – Altera o estado de Cada nó que está na Lista para o Estado escolhido.

atualizarCaso(): int – Executa as operações de finalização do caso.

enviarCaso(casoEnviado: Caso): int – Envia um Caso a todos os nós que estão alocados na classe, o iniciando.

Funções da classe FIFO: Essa classe é responsável por armazenar os casos. Ela contém uma Fila onde os casos são colocados e retirados na ordem de chegada. Abaixo estão descritas as funções mais importantes:

listarFifo(estado: ESTADO_CASO): String – Essa função devolve uma linha de texto com os dados de todos os casos que estão presentes na fila, dependendo do estado em que foi pedido.

casoTopoAtivo(lista: String): Caso – Essa função devolve o caso que está no topo de uma determinada lista, com o estado ativo, pronto para ser iniciado.

numeroNosRequisitadoTopo(lista: String): int – Essa função devolve os números de nós que é requisitado pelo próximo caso a ser iniciado de determinada fila.

Funções da classe Caso: Essa classe é responsável por armazenar os dados de cada caso. Suas funções são basicamente de acesso aos dados. Abaixo será descrita a única função com outro papel:

preencherCasoJob(): int – Essa função é responsável por pegar os dados que vieram no arquivo job e os preencher nas suas determinadas variáveis. Ele é chamado cada vez que o construtor da classe é utilizado.

A seguir, na Figura 6, é apresentado o diagrama de classes do módulo de Interface com o usuário. Esse módulo contém as funções para a execução local do arquivo job transformado em Shell Script e de monitoramento dessa execução. Também existem funções que são usadas para a comunicação com o módulo escalonador.

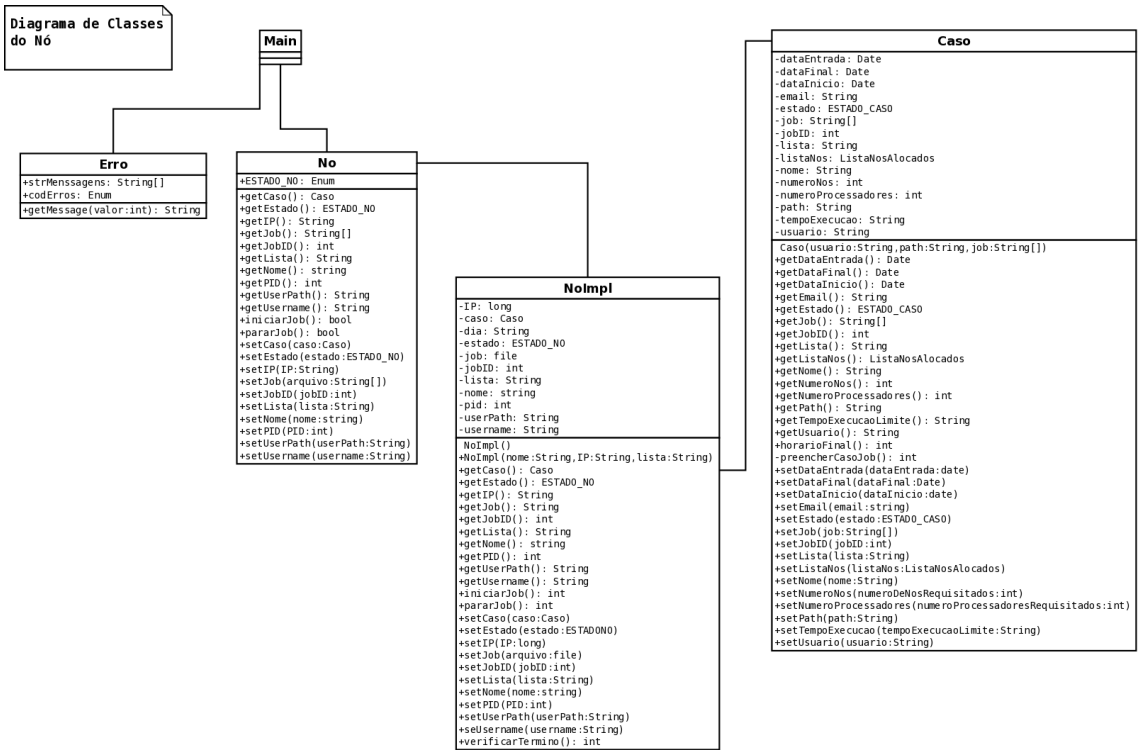

Figura 6: Diagrama de classes do módulo de interface com os nós.

As funções dessas classes já foram descritas acima, na parte de descrição das classes do módulo Escalonador. Para uma completa descrição, é necessária a leitura do JavaDocs do Módulo de Interface com os Nós, que se encontra na mídia anexa ao projeto.

Na Figura 7, é apresentado o diagrama de classes do arquivo Base.jar. Esse arquivo contém as classes que devem ser compartilhadas entre os diversos módulos. Essas classes devem ser compartilhadas em função da utilização da biblioteca RMI, que faz a exigência que todos que acessam determinada classe tenham uma cópia da interface que foi exportada, portanto, para facilitar a execução, foi criado um arquivo jar com as interfaces. As outras classes que também compõe esse pacote são necessárias para o correto funcionamento da interface, e, além disso, são compartilhadas com outros módulos do software.

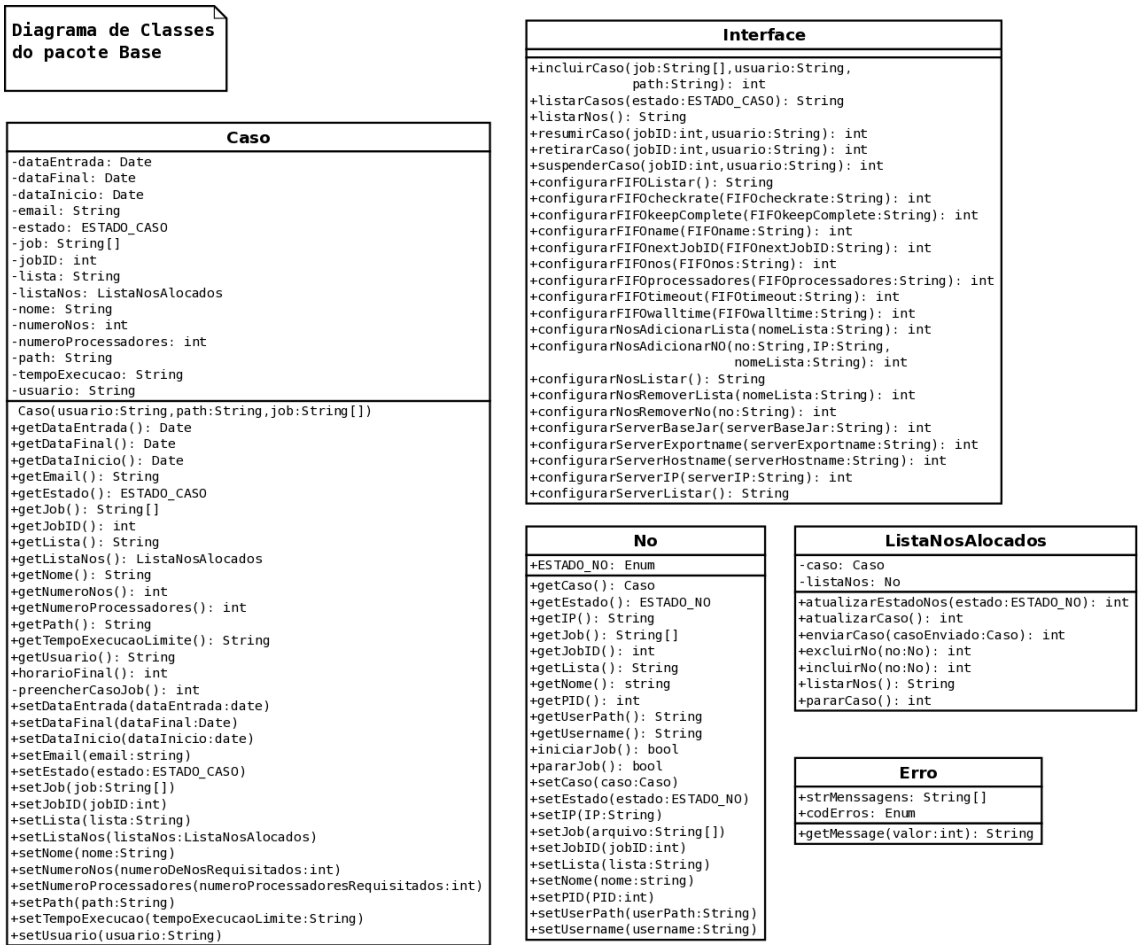

Figura 7: Diagrama de classes do base.jar

As funções dessas classes já foram descritas acima, na parte de descrição das classes do módulo Escalonador. Para uma completa descrição, é necessária a leitura do JavaDocs do arquivo Base.jar, que se encontra na mídia anexa.

5.5 – Soluções apresentadas

5.5.1 – Uso da classe Properties

O método de guardar as informações de configuração foi a classe Properties, do pacote java.util. Com essa classe, temos um acesso simples a configurações. É criado um arquivo com o nome do tipo de configuração que ele será responsável por

armazenar. Dentro desse arquivo, temos uma lista com nomes de variáveis acompanhadas do caractere de igual e seguidas do valor da variável.

Com esse arquivo pronto, o software é capaz de ler as variáveis através da função getProperty, que devolverá uma linha com o valor da variável escolhida. Para salvar um valor, o procedimento é semelhante, basta chamar a função setProperty com o nome da variável a ser guardada e o seu valor.

5.5.2 – Implementação da fila de casos.

Para a implementação da Fila de casos, foi escolhida a classe ArrayDeque, do pacote java.util. Essa classe foi escolhida para implementar a lista de casos por ser uma fila que pode ter o seu tamanho variado dinamicamente, sem restrições. Os seus métodos de acesso facilitam a criação de listas de variados estilos, sendo a do tipo fifo, onde a ordem de entrada dos objetos é a mesma ordem de saída dos mesmo, uma delas.

É importante observar o fato de que o primeiro objeto da lista não é necessariamente o próximo a ser executado. O caso deve estar com o estado ativo para ser iniciado, portanto, se um caso se encontra suspenso, mesmo estando no topo da fila ele não será o próximo a ser executado.

Durante a implementação foi escolhida a criação de somente uma fila para guardar todas as diferentes fifos pertencentes a clusters homogêneos que possam haver no sistema. Um caso estar no topo da fila do software, não necessariamente o faz ser o próximo a ser executado. O primeiro caso de cada fifo que está contido no sistema é o próximo caso que deve ser iniciado, lembrando que o estado do caso deve ser o ativo.

Em outras palavras, durante a busca na fila de casos para procurar o próximo caso que deve ser executado, a lista é varrida procurando o primeiro caso com estado ativo de cada uma das filas. Esse caso que é checado para ver se existe as condições para ser iniciado.

5.5.3 – Implementação da lista de Nós.

Para a implementação da lista de nós, foi escolhida a classe ArrayDeque, do pacote java.util. Essa classe foi escolhida para implementar a lista de nós por ser uma lista que pode ter o seu tamanho variado dinamicamente, sem restrições. A sua indexação por objeto e o uso da ferramenta iterator facilitam na hora da organização.

Sua característica de não deixar que um objeto vazio ou que cópias do mesmo objeto estejam em seu conteúdo é necessária para que um nó não seja contado como se fosse dois objetos dessa lista simultaneamente.

5.5.4 – Uso de RMI para conexão com objetos Remotos

O uso do pacote java.rmi foi escolhido por ser a maneira mais fácil de implementar o código de comunicação entre módulos disponível para Java. Através desse pacote, um objeto é colocado a disposição por um computador, ou seja, é criado um meio de acesso para esse objeto remotamente. Outra parte do software, se souber o endereço exato do objeto, pode se conectar a esse módulo que exportou o objeto, sendo capaz de fazer requisições a ele através de uma interface. Uma vez que a conexão ao objeto é realizada, torna-se transparente que o objeto que está sendo acessado é remoto.

O pacote rmi também foi escolhido por ser possível a integração com o software que vem sendo desenvolvido pelo Cepel, o Encad.

Uma das desvantagens que foi encontrada durante a fase de implementação e testes, que não é mencionada na literatura que foi lida, é o tempo de resposta acentuado quando existe a distância entre as maquinas virtuais. Sempre que existe uma ligação de rede entre o objeto e o lugar em que ele é utilizado, existe um tempo de resposta, que pode chegar a poucos segundos. Esse atraso foi observado principalmente na fase de testes com mais de uma máquina.

5.5.5 – Interface por Linha de Comando

Foi escolhida a interface de texto para a interação com o usuário. Através de argumentos no tempo de execução do módulo de comunicação com o módulo escalonador, o usuário informa o que deve ser executado. Sempre é mostrada uma mensagem informando se foi possível se conectar ao software. No caso de um comando que tenha como saída uma informação de texto, essa pode ser visualizada no terminal de onde foi chamado o comando.

5.6 – Conclusão

Neste capítulo foi discutida a análise e projeto do sistema, identificando as classes e as principais funções necessárias para a realização do projeto. Foi proposta para a modelagem do sistema a criação de duas filas, uma contendo os dados de casos a serem executados e uma contendo os dados dos locais onde esses casos devem ser executados.

A solução proposta apresenta as seguintes Características:

- Solução Orientada a Objetos;
- Estrutura em FIFO da fila de casos permite a correta ordem de processamento dos casos;
- Monitoramento através da interface de comandos foi escolhida para a facilitação da apresentação dos dados;
- Possibilidade de junção do software com um outro projeto contido no CEPEL, sendo essa feita através do uso de RMI, onde é possível o acesso simples ao software através de um outro software Java.

Capítulo 6 - Resultados

6.1 - Introdução

Neste capítulo serão apresentados os resultados finais do projeto. Também serão apresentados o método de instalação do software, a interface de configuração e a interface por onde o usuário irá submeter os comandos. Será mostrado um exemplo de como inserir um caso para o processamento e as ferramentas que podem ser usadas para monitorá-lo.

6.2 – Resultados obtidos

O principal resultado obtido com este trabalho foi a implementação de uma ferramenta para gerenciar uma fila de processamento de modelos matemáticos, a ser usada pelo Cepel. A ferramenta criada não somente processa modelos matemáticos, mas permite processar qualquer procedimento computacional, desde que esse procedimento esteja escrito na forma de um arquivo que seja reconhecido pelo software, ou seja, um arquivo escrito na linguagem Shell Script, para ser executado em um sistema operacional baseado em Linux.

Além de gerenciar, a ferramenta gera relatórios na forma de log, para que seja possível o monitoramento tanto do que os usuários submetem ao escalonador como o andamento do estado da fila. É também gerado um relatório diário como os erros que ocorreram durante o funcionamento do software.

Para alcançar os objetivos desse trabalho, se passou pelas seguintes etapas:

- Construção de um documento contendo as especificações dos requisitos de software, produzido em conjunto com os desenvolvedores do Encad;

- Elaboração de um documento contendo o projeto do software. Nessa etapa também foram elaboradas e desenhadas as classes dos três módulos que compõem o sistema;
- Implementação da solução computacional para o gerenciamento de casos.

6.3 – Instalação

A instalação do software deve acontecer de forma independente. O software é composto de três aplicativos diferentes, cada um tendo o seu papel no funcionamento do sistema.

Para entender melhor o papel de cada um, é feita uma breve descrição das três partes que compõem o sistema:

- **Interface**: Módulo usado para a comunicação com o usuário. Consiste de um arquivo jar nomeado de interfaceUsuario.jar .
- **Escalonador**: Centro do sistema. Ele que é o responsável por toda a lógica do sistema. Consiste de um arquivo jar nomeado de escalonador.jar .
- **Nó**: Usado para a comunicação entre o escalonador e um nó do cluster. Deve ser instalado em cada um dos nós. Consiste de um arquivo jar nomeado de interfaceNo.jar .

Outro arquivo necessário para todos os casos é o arquivo server.policy. Ele é responsável por controlar o acesso às interfaces RMI. Um exemplo do que deve conter esse arquivo é mostrado abaixo:

```
grant{
    permission java.security.AllPermission;
};
```

Nesse exemplo, é criada uma permissão que qualquer computador com qualquer nome e IP pode se conectar a essa interface.

Com todos os arquivos necessários em mão, e definido qual parte do sistema que irá operar em cada computador, o próximo passo é iniciar a execução dos aplicativos.

Para iniciar o módulo **Escalonador**, deve-se usar o arquivo escalonador.jar. Além desse arquivo, são necessários os arquivos de configuração do software. Eles são o fifo.cfg, o nodes.cfg e o server.cfg.

O arquivo fifo.cfg é responsável por guardar as configurações da fifo. Segue abaixo um exemplo de como deve ser o seu conteúdo:

```
#Arquivo de Configuracao. Nao editar manualmente
#Sat Feb 12 17:54:19 BRST 2011
name=FIFO
```

O arquivo nodes.cfg é responsável por armazenar o nome, o endereço e a que lista pertencem os nós que o escalonador deve tentar se comunicar. Observe a ordem em que os dados são incluídos, sendo [nome=IP\:lista_que_pertence]. Segue abaixo um exemplo desse arquivo:

```
#Arquivo de Configuracao. Nao editar manualmente
#Sat Feb 12 17:54:19 BRST 2011
node05=172.16.10.105\:default
node04=172.16.10.104\:default
node03=172.16.10.103\:default
node02=172.16.10.102\:default
node01=172.16.10.101\:default
```

O arquivo server.cfg é responsável por guardar as configurações do servidor. Segue abaixo um exemplo de como deve ser o seu conteúdo:

```
#Arquivo de Configuracao. Nao editar manualmente
#Sat Feb 12 17:54:19 BRST 2011
IP=172.16.10.100
keepComplete=2400
nucleos=1
nos=2
walltime=10\:00\:00
FIFOname=default
procesadores=1
nextJobID=100
```

checkrate=120

timeout=600

hostname=Servidor

baseJar=/Users/daniel/Dropbox/ProjetoFinal/Software/Base/dist

listanomes=default

exportName=ServidorRMI

Com esses arquivos no mesmo diretório, para iniciar o software basta executar o comando: *java -jar escalonador.jar*

Para iniciar o módulo **Nó**, deve-se usar o arquivo interfaceNo.jar. Além desse arquivo, é necessário o arquivo de configuração do software. Ele é o arquivo configuração.cfg. Segue abaixo um exemplo desse arquivo:

#Arquivo de Configuracao.

#Thu Dec 30 16:34:39 BRST 2010

IP=172.16.10.101

hostname=node01

exportname=node01

listaname=default

basejarfolder=/Users/daniel/Dropbox/ProjetoFinal/Software/Base/dist

Com esses arquivos no mesmo diretório, para iniciar o software basta executar o comando: *java -jar interfaceNo.jar*

Para iniciar o módulo **Interface**, deve-se usar o arquivo interfaceUsuario.jar. Além desse arquivo, é necessário o arquivo de configuração do software. Ele é o arquivo configuração.cfg. Segue abaixo um exemplo desse arquivo:

#Arquivo de Configuracao. Nao editar manualmente

#Tue Dec 28 17:20:01 BRST 2010

ip=172.16.10.100

servername=ServidorRMI

Com esses arquivos no mesmo diretório, para iniciar o software basta executar o comando: *java -jar interfaceUsuario.jar*

6.4 – Interface de configuração

A configuração do software é feita preferencialmente através da interface de linha de comando. Essa configuração também pode ser feita editando os arquivos manualmente, mas isso não é aconselhável, salvo nos módulos de interface com o nó e de interface com o usuário, onde somente a configuração por meio de arquivo está presente.

Antes de conhecer os comandos para editar as configurações é interessante que se conheça os parâmetros a serem alterados.

No módulo de Interface com o Usuário, temos um arquivo, o configuração.cfg, que contém as seguintes opções de configuração, com o nomes auto-explicativos:

ip=172.16.10.100

servername=ServidorRMI

No módulo de Interface com os Nós, temos um arquivo, o configuração.cfg, que contém as seguintes opções de configuração, com o nomes autoexplicativos:

IP=172.16.10.101

hostname=node01

exportname=node01

listaname=nova

basejarfolder=/Users/daniel/Dropbox/ProjetoFinal/Software/Base/dist

É bom ressaltar que o basejarfolder é o local onde se encontra o arquivo Base.jar. Esse arquivo é necessário para o correto funcionamento do sistema de RMI.

No módulo Escalonador, temos três arquivos de configuração, separados em configurações dos nós, configurações da Fifo e configurações do servidor. Os principais valores dele estão explicitados na seção 6.4. Para a configuração do Escalonador, será explicado como fazer a configuração por linha de comando.

Para listar as configurações armazenadas atualmente, o seguinte comando deve ser utilizado: *interfaceUsuario.jar –ListarConf [opção]* . Deve-se escolher uma opção

dentre as disponíveis. Elas são: Nó, Fifo e Server. Se a opção Nó for escolhida, será apresentada uma lista com os nós que devem ser buscados pelo escalonador. Se a opção Fifo for escolhida, será mostrada uma lista com as configurações da fifo. Se a opção Server for escolhida, será mostrada uma lista com os valores das configurações do Escalonador.

Para modificar as configurações de **nós**, podemos:

Incluir um novo nó:

interfaceUsuario.jar –no AdicionarNo=[NomeNo=IP=NomeLista]

Remover um nó:

interfaceUsuario.jar –no RemoverNo=[NomeNo]

Adicionar uma nova lista:

interfaceUsuario.jar –no AdicionarLista=[NomeLista]

Remover uma lista:

interfaceUsuario.jar –no RemoverLista=[NomeLista]

Para modificar as configurações da **fifo**, podemos:

Modificar o taxa de checagem de novas ações na Fifo:

interfaceUsuario.jar –FIFO checkrate=[valor em milisegundos]

Modificar o tempo em que um caso é listada apos ser concluído:

interfaceUsuario.jar –FIFO KeepComplete=[valor em hora:min:seg]

Modificar o nome da Fifo:

interfaceUsuario.jar –FIFO name=[valor]

Alterar o valor do número de identificação do próximo job a ser iniciado:

interfaceUsuario.jar –FIFO nextJobID=[valor inteiro]

Alterar o número de nós padrão:

interfaceUsuario.jar –FIFO nos=[valor inteiro]

Alterar o número de processadores padrão:

interfaceUsuario.jar –FIFO processadores=[valor inteiro]

Alterar o tempo em que um nó é considerado indisponível:

interfaceUsuario.jar –FIFO timeout=[valor em milisegundos]

Alterar o tempo padrão que um casa pode ficar sendo executado:

interfaceUsuario.jar –FIFO walltime=[valor no formato hora:min:seg]

Para modificar as configurações do **servidor**, podemos:

Modificar a localização do arquivos base.jar:

interfaceUsuario.jar -server BaseJar=[valor]

Mudar o nome que o serviço será exportado:

interfaceUsuario.jar -server ExportName=[valor]

Mudar o nome do host que exportará o serviço:

interfaceUsuario.jar -server HostName=[valor]

Mudar o IP no qual o objeto será exportado:

interfaceUsuario.jar -server IP=[valor no formato 000.000.000.000]

Ao executar o comando, as opções são salvas automaticamente. É importante observar que não existe validação dos dados, com isso, é vital observar o tipo de dado que deve ser entrado. Essa lista foi demonstra todas as configurações possíveis do software.

6.5 – Interface de uso

A interface de uso do software deve ser feita somente no módulo Interface do Usuário. Ela é feita através de linha de comando, onde os argumentos da chamada do aplicativo *interfaceUsuario.jar* vão determinar o comando a ser executado.

Os comandos que podem ser executados são basicamente os de exibição das listas de casos e das listas de nós e os comandos de manipulação da fila.

Os comandos de exibição se dividem em dois:

Listar todos os nós que estão conectados, com o seu estado;

interfaceUsuario.jar -Listar nos

Listar todos os casos que estão no escalonador;

interfaceUsuario.jar -Listar fila [opções]

As opções de listar os casos podem ser nenhuma, no caso serão exibidos todos os casos com estado ativo, ou uma das opções abaixo:

Ativo, Erro, Concluídos e Todos. Cada um irá exibir os casos com o estado correspondente.

Os comandos para a manipulação da fila são os quatro básicos, de inserir, remover, pausar e resumir. Para cada uma dessas opções segue uma descrição:

Adicionar um caso à fila:

interfaceUsuario.jar -Fila adicionar=[arquivo job]

Remover um caso da fila:

interfaceUsuario.jar -Fila remover=[JobID]

Suspender um caso da fila:

interfaceUsuario.jar -Fila suspender=[JobID]

Resumir um caso da fila:

interfaceUsuario.jar -Fila resumir=[JobID]

Com isso, todos os comandos de interface estão mostrados. É bom ressaltar que mais de uma ação pode ser executada na mesma linha de comando. Também é bom observar que o número de argumentos deve ser correto para a execução correta de cada comando. Se existirem argumentos a menos, o sistema não irá processar o comando, se existirem argumentos a mais, ou que não estejam devidamente separados pelo caractere de igual, eles não serão interpretados e ocasionarão um erro.

Abaixo, estão algumas telas do uso do software. Essas telas foram capturadas das máquinas virtuais. Primeiro, na figura 8, temos uma tela onde o software monitor está sendo executado. Depois, na figura 9, temos uma tela do software gerenciador sendo executado em outra máquina virtual. Por fim, na figura 10 e 11, temos a interface com o usuário. Na primeira tela temos a saída do comando de listar os nós. No caso da figura, temos dois nós conectados ao software. Na segunda, figura 11, temos a saída do comando listar fila, onde é exibida uma lista com todos os casos que foram concluídos recentemente, de todos os casos que estão sendo executados e de todos os casos que estão aguardando, esperando para serem executados.

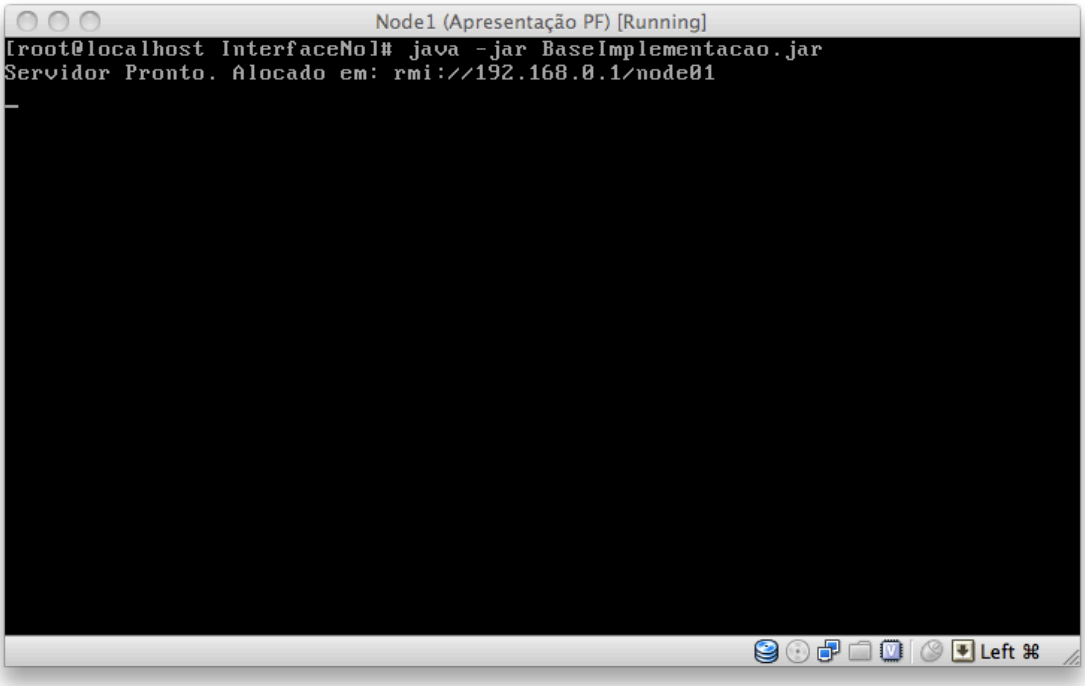

Figura 8: Nó executando o módulo Monitor.

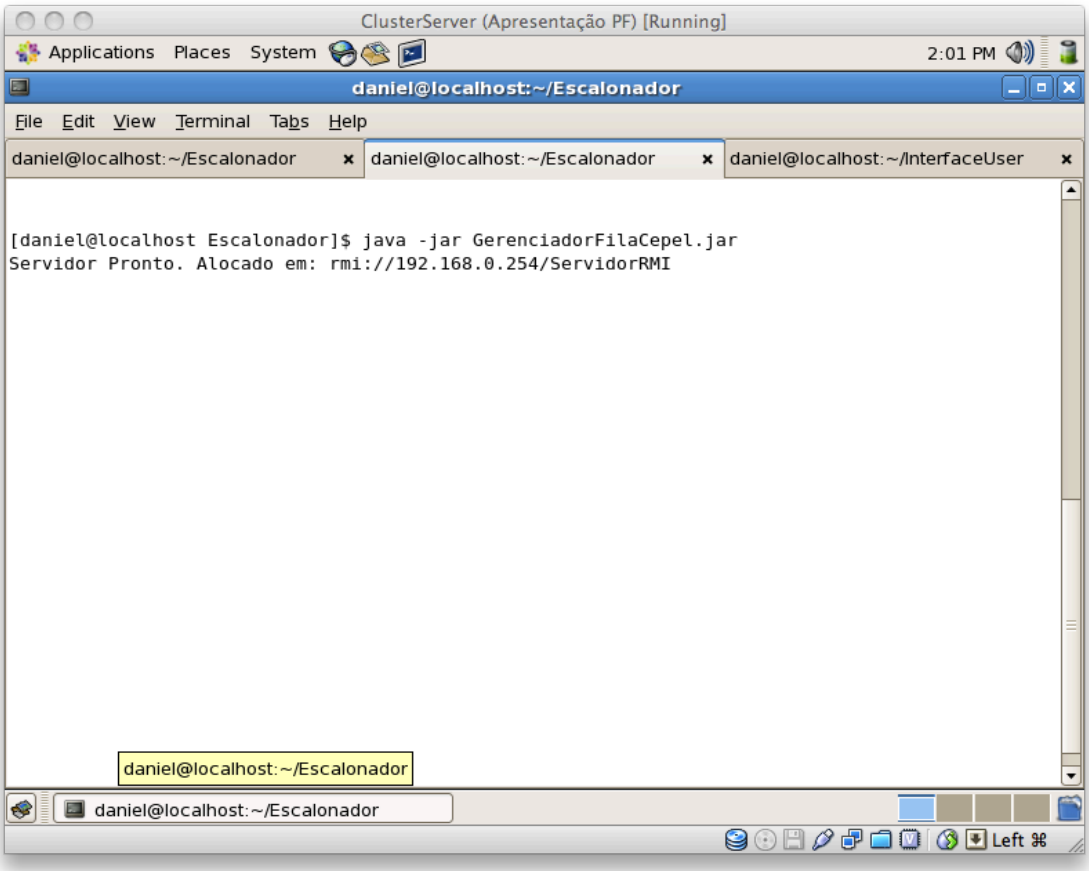

Figura 9: Módulo gerenciador sendo executado.

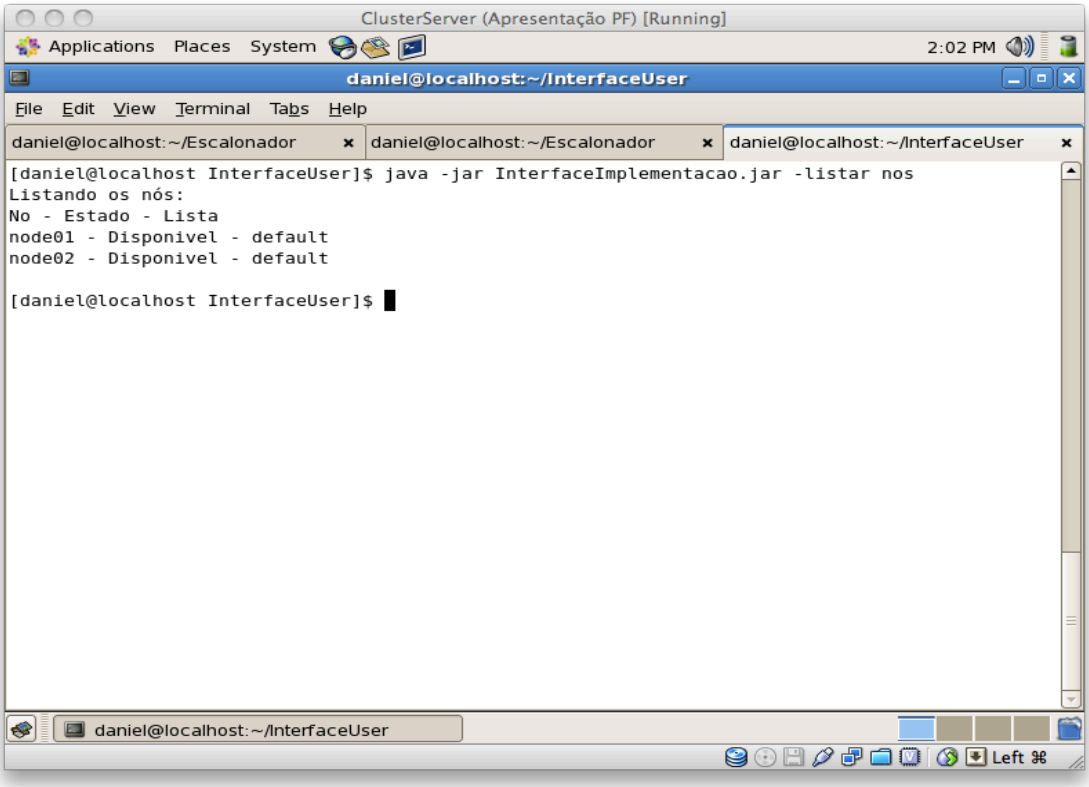

Figura 10: Módulo de interface com o usuário. Comando de listar nós.

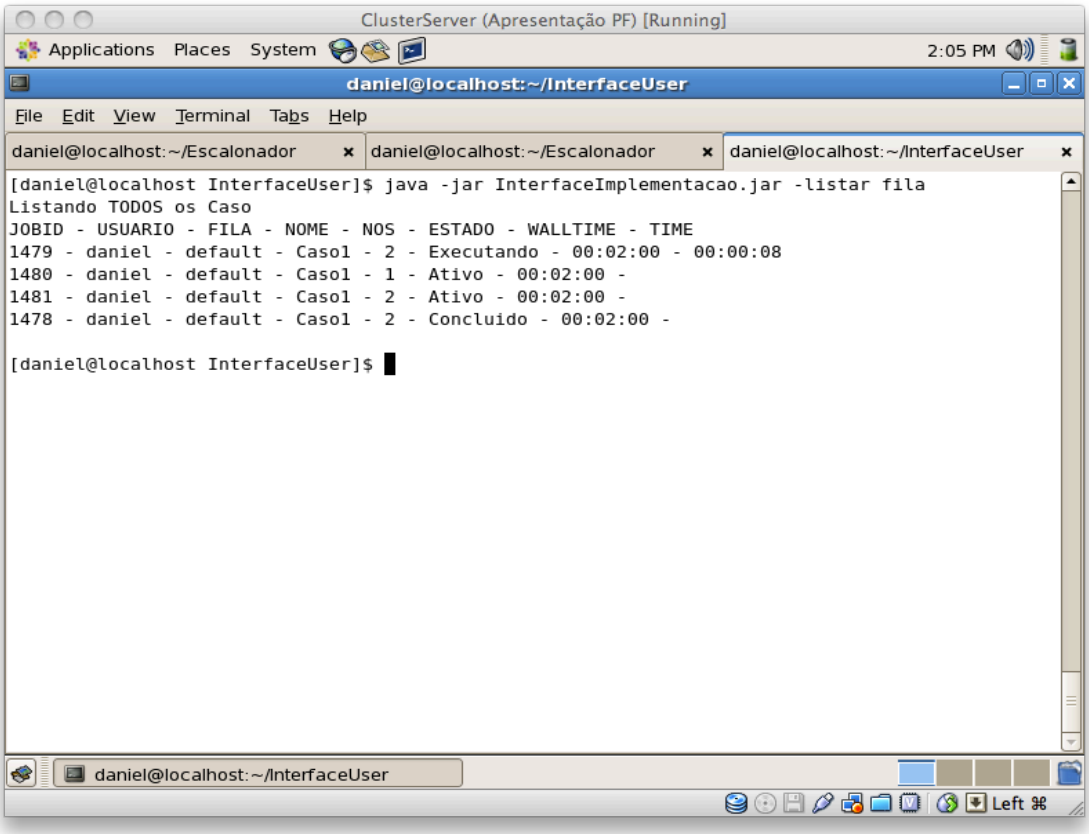

Figura 11: Módulo de interface com o usuário. Comando de listar a fila.

Capítulo 7 - Conclusão

A crescente utilização de técnicas de paralelismo para resolver problemas complexos faz com que esse trabalho tenha um importante significado. O ambiente de computação paralela é um lugar onde o controle de recursos é essencial, o desperdício de recursos computacionais pode acarretar em grandes perdas dinheiro, quando dados vitais deixam de ser processados por não ter um gerenciamento adequado de prioridade de uso de recursos.

Para diminuir o problema de ociosidade de recursos, são executados softwares que fazem o gerenciamento das atividades, tentando otimizar o número de máquinas designadas a uma tarefa e o tempo que uma tarefa deve esperar até que ela seja executada. Nesse ambiente é que entra o software que foi proposto, tentando organizar em uma fila os casos que devem ser executados no ambiente de computação paralela.

A partir dessa motivação, esse projeto tinha como objetivo criar uma ferramenta especializada para as necessidades do Cepel e de seus parceiros que usam o ambiente de programação paralela desenvolvido pelo Cepel para planejar as diferentes ações que devem ser tomadas na infra-estrutura de geração e transmissão da energia elétrica brasileira.

7.1 – Resultados

Como resultado deste projeto, foi criado um sistema para facilitar a administração de um ambiente de processamento paralelo. Junto a isso, a ferramenta possui interfaces para que seja fácil a sua futura integração com outras plataformas que necessitem utilizar o processamento paralelo, mas que não tenham a capacidade de administrar a execução dos trabalhos em cada nó individualmente. Os testes realizados em máquinas virtuais mostraram que o sistema atende a todas as funcionalidades propostas no documento de especificação de requisitos, sendo capaz de enviar e monitorar com precisão os casos que são enviados a ela, com um atraso de menos de um segundo para a aceitação dos comandos.

Durante os testes, os comandos da interface foram testados para verificar as suas funcionalidades e a sua usabilidade e se estavam sendo executados de forma correta. Também foram testados as situações extremas do software, como situações que poderiam gerar um deadlock. Foi testado também se a ordem em que os casos eram processados era a ordem correta, de acordo com os recursos disponíveis no sistema. O sistema conseguiu passar por todos esses testes, sendo que todos os erros que foram diagnosticados durante esse procedimento foram corrigdos.

Em termo de aprendizado, esse projeto proporcionou um aprofundamento nos conceitos que de engenharia de software, já que a confecção dos documentos foi uma parte essencial a produção do software o mais correto possível, sem que fosse preciso uma grande experimentação de diferentes requisitos de projeto. Outro aprendizado obtido foi o da linguagem de programação orienta a objetos Java. No início do projeto, eu não tinha familiaridade com a linguagem, após uma matéria e algum estudo de materiais indicados pelo meu orientador, consegui algum domínio na linguagem, conseguindo implementar com pouca dificuldade o software proposto.

7.2 – Métodos e técnicas aplicadas no desenvolvimento

Foram utilizados diversos métodos e técnicas durante o processo de desenvolvimento dos documentos e da implementação do software. É interessante destacar os seguintes:

- Pesquisa Bibliográfica;
- O Ciclo de vida do processo de desenvolvimento do software adotado foi o modelo em cascata, contendo fases bem definidas de especificação de requisitos, de análise, de projeto e de implementação;
- Para a identificação de requisitos, foram feitas pesquisas dos softwares já existentes no mercado, além de várias reuniões com os responsáveis pelo projeto para conseguir adequar os requisitos a realidade do Cepel e de seus parceiros. Revisões nos documentos ajudaram a aumentar a confiabilidade do projeto. Nessas revisões sempre era buscado a adequação dos requisitos, eliminando qualquer requisito potencialmente redundante, incompleto ou ambíguo;

- Na análise e projeto do sistema, utilizou-se as metodologias que foram aprendidas na matéria de engenharia de software, ou seja, foi utilizada o modelo de orientação a objeto para criar as classes que fazem parte do projeto. Durante esse passo, as notas de aula do professor Antônio Cláudio e o livro "Engenharia de Software", Sommerville, foram essenciais para o desenvolvimento dentro das normas da engenharia de software. O uso da UML durante o projeto facilitou bastante na elaboração de diagramas visuais, deixando o projeto mais organizado.

7.3 – Contribuições acadêmicas

Diferente dos outros projetos desenvolvidos durante a graduação, esse foi especialmente desafiador, já que não existia um tema fechado, e nem uma matéria responsável por ela, tudo desde o início teve que ser pensado por mim, com a ajuda dos diversos professores que procurei durante esse período. Uma dificuldade muito grande que tive foi o de controle de prazos, pois quando não existe aquele acompanhamento constante de um responsável, as tarefas acabam sendo feitas na última hora.

Fora essas dificuldades, o projeto me mostrou como é complexa a produção de um documentos de análise de requisitos com todos os requisitos necessários validados. Desde o início do projeto até o final, vários requisitos foram alterados, retirados, modificados, esquecidos. Mas essas etapas só ajudaram a criar um software melhor, com todas as características desejadas presentes.

7.4 – Trabalhos futuros

O principal trabalho futuro é a associação desse software ao software Encad do Cepel. Esse software terá toda a sua documentação e códigos fontes passados para os pesquisadores do Cepel, que provavelmente o usarão, pelo menos em parte, para complementar a ferramenta deles. Outro trabalho futuro que sempre deve estar presente

em todo software é a busca por erros no sistema e aprimoramento da performance do mesmo. Testes mais intensos devem ser feitos, de preferência em ambientes reais, ou seja, seria necessário que um cluster ficasse disponível para testes mais amplos, com dados reais.

Bibliografia

DIAS, Daniel Souza, *Especificações de Requisitos de Software do Gerenciador de Filas de Processamento de Dados para o Cluster do CEPEL*, 2010.

DIAS, Daniel Souza, *Projeto do Softwareo Gerenciador de Filas de Processamento de Dados para o Cluster do CEPEL*, 2010.

SOMMERVILLE, IAN, *Engenharia de software*, 8ª edição. Ed. Pearson Addison-Wesley, 2007.

JOSÉ PINTO, Roberto; SILVA DUARTE, Vitor; EMANOEL QUADROS, André; Implementação de uma Metodologia para Execução do Programa NEWAVE num Ambiente de Processamento Distribuído. CEPEL.

TORQUE Admin Manual: TORQUE Resource Manager version 2.5.0. Disponível em: <http://www.clusterresources.com/products/torque/docs/> Acesso em: 3 de setembro 2010.

HORSTMANN, Cay S. Big Java. Porto Alegre: Bookman, 2004.

GROPP, William; MPICH2 User's Guide. Disponível em: <http://www.mcs.anl.gov/research/projects/mpich2/documentation/files/mpich2-1.3.2-userguide.pdf>. Acesso em: 3 de setembro de 2010.

BAYUCAN, Albeaus. *Portable Batch System Administrator Guide*. Disponível em: <http://lsec.cc.ac.cn/chinese/lsec/doc/v2.3_admin.pdf>. Acesso em: 3 de setembro de 2010

BAYUCAN, Albeaus. *TM PBS Pro User Guide.* Disponível em: <http://www.raunvis.hi.is/~finnboo/bjolfur/pbs_user_guide.pdf>. Acesso em 3 de setembro de 2010.

Apêndice A - Especificação de requisitos de software

No anexo A, Especificação de requisitos de software, é apresentado o documento de especificação de requisitos do sistema. Esse documento tem o seu próprio índice e a sua própria numeração.

Responsável pela informação:
Daniel de Souza Dias

Relatório de Mudanças

Título do Projeto	Gerenciador de Filas de Processamento de Dados para o Cluster do CEPEL.
Número da Versão	1
Data da Liberação	20/08/2010
Descrição Sucinta da Liberação	Criação do Documento

Apêndice B – Projeto de Software

No anexo B, Projeto, é apresentado o documento de projeto do sistema. Esse documento tem o seu próprio índice e a sua própria numeração.

Responsável pela informação:
Daniel de Souza Dias

Relatório de Mudanças

Título do Projeto	Gerenciador de Filas de Processamento de Dados para o Cluster do CEPEL.
Número da Versão	1
Data da Liberação	25/10/2010
Descrição Sucinta da Liberação	Criação do Documento