NEMÉSIO FREITAS DUARTE FILHO

Implantação do *middleware Globus Toolkit 4* para aplicações em ambientes de *Grid*

Monografia de Graduação apresentada ao Departamento de Ciência da Computação da Universidade Federal de Lavras como parte das exigências do curso de Ciência da Computação para obtenção do título de Bacharel em Ciência da Computação.

Lavras
Minas Gerais - Brasil
2008

NEMÉSIO FREITAS DUARTE FILHO

Implantação do *middleware Globus Toolkit 4* para aplicações em ambientes de *Grid*

Monografia de Graduação apresentada ao Departamento de Ciência da Computação da Universidade Federal de Lavras como parte das exigências do curso de Ciência da Computação para obtenção do título de Bacharel em Ciência da Computação.

Área de Concentração:
Processamento Paralelo e Distribuído

Orientadora:
Profa. Dr. Marluce Rodrigues Pereira

Lavras
Minas Gerais - Brasil
2008

.

Ficha Catalográfica preparada pela Divisão de Processos Técnicos
da Biblioteca Central da UFLA

Filho,Nemésio Freitas Duarte

Implantação do *middleware Globus Toolkit 4* para aplicações em ambientes de *Grid* /Nemésio Freitas Duarte Filho. Lavras - Minas Gerais, 2008. 79p : il.

Monografia de Graduação - Universidade Federal de Lavras. Departamento de Ciência da Computação.

1. *Grid*. 2.*Globus Toolkit* 3. *Middleware*. I. FILHO, N. F.D. II. Universidade Federal de Lavras. III. Título.

NEMÉSIO FREITAS DUARTE FILHO

Implantação do *middleware Globus Toolkit 4* para aplicações em ambientes de *Grid*

Monografia de Graduação apresentada ao Departamento de Ciência da Computação da Universidade Federal de Lavras como parte das exigências do curso de Ciência da Computação para obtenção do título de Bacharel em Ciência da Computação.

Aprovada em *17 de junho de 2008*

Prof. Dr. Cláudio Fabiano Mota Toledo

Prof. Dr. Wilian Soares Lacerda

Profa. Dr. Marluce Rodrigues Pereira
(Orientadora)

Lavras
Minas Gerais - Brasil

Dedico esse trabalho a duas pessoas que foram muito especiais na minha vida, Manuel Laginha e Luiz Carlos de Paula Mello, duas pessoas que sempre foram motivo de orgulho para mim, mesmo estando longe permanentemente, sempre me espelharei em seus atos. Pois verdadeiros homens são lembrados pelos seus atos e pelas suas atitudes.

Agradecimentos

Agradeço aos meus pais, Analice de Paula Mello Duarte e Nemésio Freitas Duarte, e toda minha família pelo apoio que me deram ao longo destes quatro anos e meio e terem me dado todas as oportunidades para concluir um curso de ensino superior. Agradeço também aos meus amigos e a minha namorada Amanda Soares Giroto que sempre estiveram ao meu lado me incentivando, e a duas pessoas que fizeram parte de minha vida ao longo destes quatro anos e meio de faculdade Jaime Daniel Corrêa Mendes e Nestor Vicente Soares Netto, e a minha orientadora Marluce Rodrigues Pereira por ter me auxiliado e contribuído para a conclusão deste trabalho.

Implantação do *middleware Globus Toolkit 4* para aplicações em ambientes de *Grid*

Resumo

A computação em *Grid* consiste em um conjunto de serviços, protocolos e *softwares* que integra recursos computacionais distribuídos em um ambiente unificado. É considerada uma solução para o problema enfrentado pelas organizações, no processamento e armazenamento de uma quantidade cada vez maior de dados.

Middleware específicos são usados para a construção de um ambiente de *Grid*. Porém um dos problemas no projeto de um *middleware* para *Grid* é a sua complexidade, pois devem ser considerados o *hardware, os softwares*, a autenticação, a segurança, a comunicação entre os recursos computacionais geograficamente distantes e tudo mais que envolve este ambiente amplo e geograficamente distribuído. Neste ambiente a maior dificuldade enfrentada por um usuário é a submissão de tarefas e o acompanhamento da execução e geração de arquivos de saída.

Este trabalho aborda as características e componentes de um ambiente de *Grid*, apresentando algumas ferramentas existentes e focando no *middleware Globus Toolkit 4*.

Palavras-Chave: *Globus Toolkit; Grid; middleware*.

Deployment of Globus Toolkit 4 middleware for applications in Grid environments.

Abstract

Grid computing consists in a range of services, protocols and softwares which incorporate distributed computing resources in an unified environment. It is considered as a solution to the problem faced by organizations to process and store the increasing amount of data.

The use of middlewares is necessary when a Grid environment is created. However the project of a middleware for Grid is not simple, because should be considered everything involving this large and geographically distributed environment: hardware, software, authentication, security, communication between the computational resources geographically distant, etc. Usually, when an user uses a Grid environment, he finds some difficulty to submit tasks, monitoring the execution and the generated files. So, the Grid middlewares are associated to Grid portals to help users.

This work addresses the characteristics and components of Grid environments, presenting some existing tools and focusing on the Globus Toolkit 4.

Keywords: Globus Toolkit; Grid; middleware.

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

INTRODUÇÃO

Atualmente, o volume de dados gerado por pesquisas científicas e por grandes empresas é cada vez maior e pode estar armazenado em locais geograficamente distantes. Através da *Internet* esta informação pode ser facilmente acessada. Porém, surge o problema de como filtrar o conjunto de informações de forma que se consiga obter dados relevantes. Para isso, a capacidade de armazenamento e principalmente de mineração destes dados precisa acompanhar o crescimento do volume de informação.

Outro fato que deve ser considerado é que o *hardware* teve uma redução de seus custos nos últimos anos. Hoje em dia, adquirir um computador *desktop* chega a custar apenas algumas centenas de dólares, diferente dos milhares que custava alguns anos atrás. Desta forma, atualmente há muitos computadores que passam a maior parte do tempo ociosos, seja por estarem desativados ou por estarem sendo usados apenas como ferramentas de execução de textos ou navegação na *web*. Estes computadores poderiam ser utilizados para executar informações quando estivessem ociosos, aproveitando um recurso computacional já existente.

A tecnologia de *Grids* Computacionais surgiu como uma solução a este problema. A idéia central dos *Grids* é usar a capacidade ociosa dos computadores para executar dados ou armazená-los. Desta forma, uma empresa ou uma instituição de ensino ou pesquisa pode empregar toda a capacidade de seus recursos computacionais, que incluem aplicações, servidores, bases de dados, dispositivos de armazenamento, processadores de alta capacidade, entre outros, tratando todos esses recursos como um serviço virtualizado. Assim sendo, o serviço será fornecido de forma confiável e clara, independentemente da localização do computador que executou ou armazenou uma determinada tarefa ou dado.

Segundo Foster et al [27], *Grid computing*, ou computação em *Grid*, ou computação em Grade é um modelo computacional capaz de alcançar uma alta taxa de execução de tarefas dividindo-as entre diversos computadores, que podem estar em uma rede local ou em uma rede de longa distância, formando uma máquina virtual. Estas tarefas serão executadas no momento em que os computadores não estiverem sendo utilizados pelo usuário, evitando o desperdício de ciclos de CPU da máquina utilizada.

No contexto deste trabalho será utilizado o termo computação em *Grid* .

Este trabalho tem como objetivo configurar um ambiente de *Grid* utilizando ferramentas para ambiente de *Grid* já existentes. Tais ferramentas visam facilitar a utilização de um ambiente de *Grid* por um usuário que deseja submeter suas tarefas para serem executadas neste ambiente. Por isso, fornecem facilidades de comunicação entre os recursos computacionais, monitoramento destes recursos e das tarefas a eles submetidas, autenticação e segurança para o acesso ao ambiente, entre outros serviços. Durante o desenvolvimento deste trabalho foram estudadas as ferramentas existentes para tal objetivo. Após a análise das ferramentas existentes, foi definido que o *middleware Globus Toolkit 4*, o gerenciador de *jobs PBS* e o *framework* para desenvolvimento de portais para *Grid, Gridsphere*, seriam os mais adequados. Estas ferramentas foram instaladas, configuradas e testadas para um ambiente de teste e mostraram bons resultados quanto à usabilidade e segurança.

O trabalho também mostra as dificuldades da implantação da tecnologia *Grid*. Mesmo tendo um grande poder computacional esta tecnologia é nova e apresenta dificuldades em sua implantação.

Os demais capítulos deste texto estão organizados da seguinte forma. O capítulo 2 apresenta o referencial teórico no qual este trabalho foi embasado. O capítulo 3 aborda a metodologia utilizada juntamente com os métodos utilizados e as instalação feitas no ambiente de *Grid*. O capítulo 4 apresenta os resultados obtidos, os testes efetuados no ambiente e os principais erros e dificuldades encontrados na implantação. E o capítulo 5 traz a conclusão do trabalho juntamente com possíveis trabalhos futuros a serem realizados.

Capítulo 2

REFERENCIAL TEÓRICO

Este capítulo apresenta os conceitos existentes na literatura sobre computação em *Grid*, os serviços necessários para realização de tal computação e as principais ferramentas existentes para prover tais serviços.

2.1 Computação em *Grid*

Na literatura, existem diferentes definições para computação em *Grid*. Esta seção visa apresentar alguns destes conceitos, benefícios e componentes necessários para esta computação.

Um dos conceitos de computação em *Grid* é que usuários separados geograficamente podem compartilhar, de forma dinâmica, recursos computacionais [32]. Para Berstis [21] a definição de computação em *Grid* seria uma evolução da computação distribuída. O objetivo é criar uma ilusão de um computador virtual, de fácil acesso e com um grande poder computacional e dispositivos sendo compartilhados. Os *Grids* removem as conexões fixas entre aplicações, servidores, bases de dados, máquinas, armazenamento, entre outros, tratando tudo como um serviço virtualizado [34]. Assim, os recursos computacionais podem estar em um mesmo ambiente ou alocados em diferentes locais geograficamente distantes.

A principal diferença entre um *cluster* e um *Grid* é que um *cluster* possui um controlador central, um único ponto de onde é possível utilizar todo o poder de processamento do *cluster*. Os nós de um *cluster* encontram-se em um mesmo local.

Os *Grids* por sua vez são uma arquitetura mais democrática onde embora possa existir algum tipo de controle central, temos um ambiente fundamentalmente cooperativo, onde empresas, universidades ou mesmo grupos de usuários compartilham os ciclos ociosos de processamento em seus sistemas em troca de poder utilizar parte do tempo de processamento do *Grid*.

Por exemplo, duas empresas sediadas em países com fusos-horário diferentes poderiam formar um

Grid, combinando seus servidores *web*, de modo que uma possa utilizar os ciclos de processamento ociosos da outra em seus horários de pico, já que com horários diferentes os picos de acessos aos servidores de cada empresa ocorrerão em horários diferentes.

Outra idéia bastante defendida é que a computação seja confiável e transparente. Deste modo, não importa onde seus dados ou sua aplicação residam ou qual computador processa sua requisição. O usuário requisita um dado ou processamento e ele é entregue, independente de onde esteja ou quando o solicite. É análogo à utilização da energia elétrica. Quando um interruptor é acionado por um usuário, não se sabe qual gerador ou hidrelétrica está lhe provendo a energia, mas esta lhe é entregue de maneira imediata e transparente. Na verdade, foi desta analogia que partiu a nomenclatura *Grid*, baseado nas malhas de interligação dos sistemas de energia elétrica [39].

De acordo com Foster *et al* [27], todos estas informações e conceitos sobre a tecnologia *Grid* são antigas. Esta tecnologia vem sendo estudada desde 1965 em faculdades norte americanas, onde se estudavam e desenvolviam a computação em vários dispositivos computacionais, porém somente nos dias atuais, com os desenvolvimentos da computação é que se pode colocar em prática todos esses conceitos da tecnologia *Grid*.

Analogamente à *internet*, a computação em *Grid*, desenvolveu-se nas comunidades acadêmica e de pesquisa. Entre os fatores que influenciaram seu desenvolvimento destacam-se:

- As pesquisas envolviam um grande número de professores e estes geralmente, encontravam-se em locais afastados, por isso notou-se a urgência de se criar um ambiente computacional para compartilhar recursos e resultados dinamicamente;
- Também era importante escalar facilmente, para poder acomodar uma quantidade cada vez maior de dados e poder computacional;
- Manter os custos baixos.

Esses requisitos são resolvidos pela arquitetura dos *Grids*. Estes podem escalar de maneira fácil e dinamica, podem englobar máquinas localizadas em lugares diferentes, utilizando seus recursos e tempo de processamento ociosos, e podem utilizar *hardware* comum, não necessitando da utilização de máquinas de grande porte como supercomputadores.

Deste modo, seu aproveitamento no meio acadêmico foi de grande importância e de grande utilidade em diversos projetos. O projeto *SETI@home, Search for Extraterrestrial Intelligence* [40], é um dos projetos de maior referência sobre *Grids* no meio científico. Os dados colhidos de sinais de telescópios, receptores de rádio e outras fontes de monitoramento eram distribuídos pelos computadores, através da *internet*. Para serem processados estes computadores executavam os dados em busca de sinais que pudessem comprovar a existência de seres extra terrestres, aproveitando seu tempo ocioso de execução.

Para exemplificar melhor o significado, Bombonato [22] definiu três características básicas para que um sistema computacional possa ser denominado de *Grid*:

1. Recursos coordenados que não se sujeitam a um controle centralizado: Sistemas em *Grid* podem englobar recursos entre os mais variados tipos, desde o *desktop* de um usuário até um supercomputador;

2. Empregar normas, interfaces e protocolos de propósito geral: O aproveitamento de protocolos e normas é essencial para que os sistemas em *Grid* possam realizar funções fundamentais como autenticação, autorização, descobrimento de recursos e acesso a eles;

3. Prover o mínimo em qualidade de serviços, como segurança, tempo de resposta e disponibilidade.

Em [22] , Ian Baird fala que a admissão em massa da computação em *Grid* deve acontecer em três etapas:

- Primeira Etapa: *Enterprise Grids*. Ocorre, com empresas que possuam filiais em diversas cidades, países ou sedes, compartilhando recursos, dentro do domínio da empresa, atrás dos limites do *firewall*. Assim, múltiplos projetos ou departamentos dentro da organização podem compartilhar seus recursos computacionais, aproveitando-os de forma otimizada em tarefas como engenharia colaborativa, mineração em grandes bases de dados e renderização de *frames* para animação;

- Segunda Etapa: *Partner Grids*. Uma expansão e interligação entre *Grids* de organizações de áreas de interesse e pesquisa parecida. Empresas que trabalham no ramo de farmácias, por exemplo, tornando-se parceiras e partilhando meios para atingir um objetivo em comum;

- Terceira Etapa: *Service Grids*. Os sistemas em *Grid* se tornarão um sistema utilitário, com usuários aproveitando recursos computacionais sem saber onde estão localizados ou a que empresa pertencem, apenas pagando pelo uso destes.

Os benefícios que são oferecidos pela utilização da computação em *Grid* são a motivação para que estas etapas ocorram.

2.1.1 Benefícios

Um dos principais benefícios da computação em *Grid* é que recursos computacionais disponíveis e espalhados dentro de uma empresa poderão ser melhor utilizados para atender as demandas computacionais da organização.

A tecnologia *Grid* vem para revolucionar a utilização dos computadores juntamente com a computação de dados. Tendo a computação em *Grid* em crescimento, cria-se uma grande expectativa em

relação ao que se pode ou não fazer com ela. Berstis [21] afirma que a tecnologia *Grid* possibilita: explorar recursos, possui capacidade de execução em paralelo, possui dispositivos e organizações virtuais e apresenta confiabilidade.

1. Explorar recursos:

   Além dos recursos para execução de *jobs*, muitas máquinas também possuem seus discos rígidos sendo utilizados para armazenamento de dados. Assim, o *Grid* pode ser utilizado como uma alocação de espaço disponível como se fosse um disco apenas. Outra forma de alocar o espaço seria segmentar os dados de forma que as aplicações possam ser executadas em uma máquina mais próxima de onde se encontram os dados que executa, ou para garantir uma maior disponibilidade caso alguma máquina falhe.

2. Capacidade de execução em paralelo:

   Possibilidade de agilizar a execução de aplicações científicas, financeiras, processamento de imagens e simulações, no aproveitamento de execução paralela.

3. Dispositivos e organizações virtuais:

   A colaboração entre os mais diversos tipos de usuários e aplicações é outra capacidade que pode ser desenvolvida com o advento dos *Grids*. Recursos e máquinas podem ser agrupados e trabalharem juntos formando o que pode ser denominado de uma Organização Virtual (OV).

   Uma OV é uma entidade que compartilha recursos através do *Grid* utilizando uma determinada política. Comparando-se com a *internet*, seria semelhante a um *site*, mas com a diferença de poder fornecer serviços solicitados pelos usuários [39].

4. Confiabilidade:

   Existem diversas maneiras de aumentar a confiabilidade em um sistema computacional. Processadores e discos são duplicados, de modo que caso um falhe o outro assuma seu lugar. Fontes de energia e circuitos redundantes, geradores elétricos, entre outros... Todas estas formas comprovadamente aumentam a disponibilidade e confiança em um sistema, mas seus altos custos podem torná-las impraticáveis.

Os benefícios da computação em *Grid* atingirão diversas áreas de pesquisa, tais como física, bioinformática, otimização, entre outras, além de permitir a realização de grandes pesquisas experimentais nas áreas ambiental, de treinamento e educação [38]. Os problemas tratados por estas áreas de pesquisa envolvem o consumo de muitos ciclos de processamento e/ou geram várias bases de dados independentes que precisam ser integrados para que pequisadores de instituições dispersas ao redor do mundo possam realizar análises. [38]

2.1.2 Componentes

A computação em *Grid* possui dois componentes principais [23]: recursos utilizados e arquitetura.

Os recursos computacionais encontrados em um ambiente *Grid* são recursos que podem ser compartilhados e envolvem computação, armazenamento, comunicação, software e licenças.

- Computação:

  O recurso mais comum é a computação fornecida pelas máquinas do *Grid* . Os processadores podem variar em velocidade, arquitetura, software, plataforma, e outros fatores associados, tais como a memória, armazenamento e conectividade. Existem três maneiras principais de explorar os recursos de computação de um *Grid*. A primeira e mais simples é usá-lo para executar uma aplicação existente em uma máquina disponível do *Grid* , ao invés de executar em uma máquina localmente.

- Armazenamento:

  O segundo recurso mais comum de um *Grid* é o armazenamento de dados. O *Grid* proporciona uma visão integrada de armazenamento de dados. Cada máquina na rede normalmente fornece uma quantidade de armazenamento para utilização do *Grid* , mesmo que temporária. O aproveitamento do espaço de armazenamento de cada máquina pelo *Grid* é uma forma interessante de compartilhamento deste recurso.

- Comunicações:

  A velocidade das comunicações pode influenciar na execução de diversas tarefas. Através do *Grid* pode-se compartilhar conexões, tanto internas quanto externas. Algumas aplicações precisam executar um grande número de dados e estes podem não se encontrar na máquina onde está sendo executada.

- *Software* e licenças:

  Alguns *softwares* possuem licenças caras e seria impraticável sua instalação em todas as máquinas do *Grid* . Assim, requisições para empregar estes softwares seriam direcionadas para as máquinas que os possuem instalados. Deste modo, pode-se realizar uma melhor utilização das licenças adquiridas.

Baker [31] destaca quatro aspectos principais que caracterizam um *Grid* : múltiplos domínios administrativos e autonomia, heterogeneidade, escalabilidade e dinamicidade ou adaptabilidade. Múltiplos Domínios Administrativos e Autonomia: Recursos da grade são geograficamente distribuídos por múltiplos domínios administrativos e pertencem a diferentes organizações. Heterogeneidade: Uma

grade envolve uma multiplicidade de recursos que são heterogêneos em natureza e irão envolver uma vasta variedade de tecnologias.Escalabilidade: Uma grade pode crescer de alguns recursos integrados à milhões. Isso pode gerar o problema de queda de desempenho potencial à medida que o *Grid* aumenta.Dinamicidade ou Adaptabilidade: Em uma grade, falha de recurso é mais regra do que a exceção. De fato, com tantos recursos na grade, a probabilidade de algum recurso falhar é alta.

Estes aspectos geram a necessidade de *middlewares* que facilitem a criação de um ambiente em *Grid* .

- Um *middleware* de baixo nível que deve oferecer um ingresso seguro e claro aos recursos;
- Um *middleware* de nível de usuário e ferramentas para desenvolvimento de aplicações;
- Permite o desenvolvimento e otimização de aplicações distribuídas que tirem vantagem dos recursos disponíveis e da infra-estrutura.

Foster *et al* [27] define a arquitetura de um *Grid* na forma de uma pilha de protocolos, semelhante à pilha de protocolos TCP/IP, como ilustra a Figura 2.1

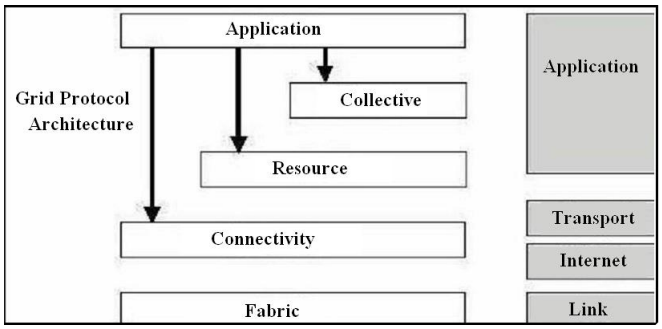

Figura 2.1: Arquitetura de um *Grid* comparado com a arquitetura TCP/IP [27].

- *Fabric*: Interfaces para controle local.

  Esta camada fornece funcionalidades com as quais o compartilhamento de recursos pelo *Grid* torna-se possível. Implementa operações locais e específicas para cada tipo de recurso compartilhado pelo *Grid* , fornecendo suporte às funções das camadas superiores.

- *Connectivity*: Comunicação fácil e segura.

  A camada *Connectivity* define os protocolos de comunicação e de segurança necessários para transferências de rede específicas dos *Grid*, permitindo transferência de dados entre recursos, utilizando as funcionalidades da camada *Fabric*.

As necessidades supridas pela camada incluem transporte, roteamento e nomeação. Atualmente estes protocolos são mapeados para a pilha de protocolos TCP/IP, especificamente *internet* (IP e ICMP), transporte (TCP e UDP) e aplicação (DNS, OSPF).

- *Resource*: Compartilhado recursos simples

  A camada *Resource* utiliza as funcionalidades da camada *Connectivity* para a negociação segura, monitoramento, controle e contabilização de operações de compartilhamento em recursos individuais.

- *Collective*: Coordenando múltiplos recursos.

  Diferentemente da camada *Resource*, que interage com um recurso simples, a camada *Collective* fornece protocolos e serviços que não estão associados a um recurso específico e sim a coleções destes. Para isto, ela disponibiliza algumas facilidades de compartilhamento:

- *Applications*: A camada final da arquitetura *Grid*.

  Compreende as aplicações de usuário que operam no ambiente de um *Grid*. As aplicações são construídas através da utilização de serviços providos por cada camada. Em cada uma destas, existem protocolos definidos que fornecem serviços como gerenciamento e localização de recursos, acesso a dados, entre outros.

2.2 Ferramentas de Desenvolvimento

Existem algumas ferramentas que facilitam a utilização da tecnologia de *Grid* . Dentre estas ferramentas, destacam-se o Legion [30], que é uma aplicação comercial paga e o *Globus Toolkit* [28], que é um *middleware* desenvolvido por uma comunidade que adota o *software* livre, disponibilizando esta ferramenta via *internet*. Além destas, existem também o Glite, Easygrid entre outras.

O Glite é um *middleware* para computação em *Grid* que surgiu de esforços da cooperação de mais de 80 pessoas de 12 diferentes centros de pesquisa acadêmicos e industrial como parte do projteo EGEE (*Enabling Grids for E-sciencE*) [5]. O Glite proporciona um *framework* para construir aplicações para serem processadas usando o poder computacional e os recursos de armazenamento distribuído pela *internet*. Ele é voltado para a construção de diferentes serviços que sejam de fácil instalação e configuração na plataforma escolhida.

Easygrid [3] é um *framework* para execução de programas MPI em ambiente de *Grid*. O objetivo do framework é gerar aplicações capazes de executar em ambientes dinâmicos, instáveis e distribuídos oferecido por *Grids* computacionais. Para isso foram implementados algoritmos de escalonamento de tarefas estáticos e dinâmicos.

Dentre estas ferramentas, o *Globus Toolkit* e o Legion são os mais utilizados atualmente, porém o Legion é um *software* pago e apresentas algumas características diferentes do *Globus Toolkit*. O foco deste trabalho será em cima do *Globus Toolkit* e mostraremos as diferenças entre os dois..

2.2.1 *Globus Toolkit*

O *Globus Toolkit* (GT) [28] foi criado na década de 90 para suportar sistemas desenvolvidos em arquiteturas orientadas a serviço em ambiente de computação em *Grid*.

Os componentes do GT suportam diversas funcionalidades para o bom funcionamento de um *Grid*, como a descoberta e gerenciamento de recursos, movimentação de dados, monitoramento do ambiente e questões de segurança [26].

Em 2005, foi lançada a versão 4 do GT (GT4), desenvolvida pela *Globus Alliance*, que representa um grande avanço na implementação de *Web Services* de forma padronizada e integrada. *Web Service* é um serviço proposto para *internet* que utiliza um sistema de mensagens padronizado [24]. A comunicação entre diferentes aplicações é feita através da troca de mensagens no formato XML *(eXtensible Markup Language)*. Essa padronização é necessária para a solução proposta pelo *Web Service*, possibilitando uma troca de mensagens entre aplicações desenvolvidas em linguagens de programação diferentes e em alguns casos sendo executadas em sistemas operacionais diferentes.

A construção do GT4 foi baseada nas lições aprendidas nas versões anteriores e motivada por projetos que funcionavam em ambiente de *Grid*, como o projeto *Earth System Grid* no qual vários laboratórios acessam remotamente o sistema via portal *web* [26].

A padronização do GT4 facilitou a construção e entendimento de cada componente do *Globus*, fazendo uso dos mecanismos de *Web Service* para definir interfaces e estruturas de comunicação. O uso de documentos em padrão XML define a maioria dos principais componentes, mas ainda existem nessa versão mecanismos que não utilizam *Web Services*, como o *GridFTP* (que auxilia na transferência de arquivos no ambiente *Grid*) e a *SimpleCA* (serviço para a autenticação e segurança do ambiente), devido ao seu baixo desempenho.

O GT4 possui componentes que gerenciam a infra-estrutura, disponibilizando os elementos do *Grid* que cada aplicação irá usufruir. Componentes do GT4 como *Grid Resource and Management* (GRAM), *Reliable File Transfer* (RFT) e *GridFTP* são usados para gerenciamento computacional e ativação de elementos, gerenciamento de transferência de arquivos e transferência de dados, respectivamente. As etapas de descoberta e monitoração também são controladas pelo GT4 e são funções essenciais para o bom funcionamento do *Grid*.

A monitoração consiste em detectar e diagnosticar problemas que ocorrem nos serviços do ambiente e a descoberta permite identificar os serviços com propriedades desejadas. Ambas tarefas coletam

informações de várias fontes de informação no ambiente de *Grid*.

É fato que a segurança é um aspecto necessário em qualquer ambiente computacional. Mas mecanismos de segurança tornam-se ainda mais necessários em ambientes de *Grid*, onde as máquinas não estão centralizadas, dificultando o aproveitamento de mecanismos físicos de segurança.

No GT4, existem componentes que implementam credenciais e protocolos que endereçam mensagens de proteção, autenticação e autorização. Cada nó do *Grid* possui sua credencial, que deve ser trocada entre as máquinas autenticando-as no ambiente.

Um grande e complexo problema das aplicações em *Grid* é o ingresso a uma grande quantidade de dados em diferentes lugares. No GT4, existem componentes com a intenção de solucionar estes problemas. Um exemplo é o *GridFTP* que transfere grande quantidade de dados com confiança, segurança e alta desempenho [20].

2.2.2 Arquitetura do *Globus Toolkit* 4

A arquitetura do GT4 é formada por componentes WS (*Web Service*) e não-WS, conforme apresenta a Figura 2.2, onde os componentes WS estão à esquerda da figura e componentes não-WS estão à direita.

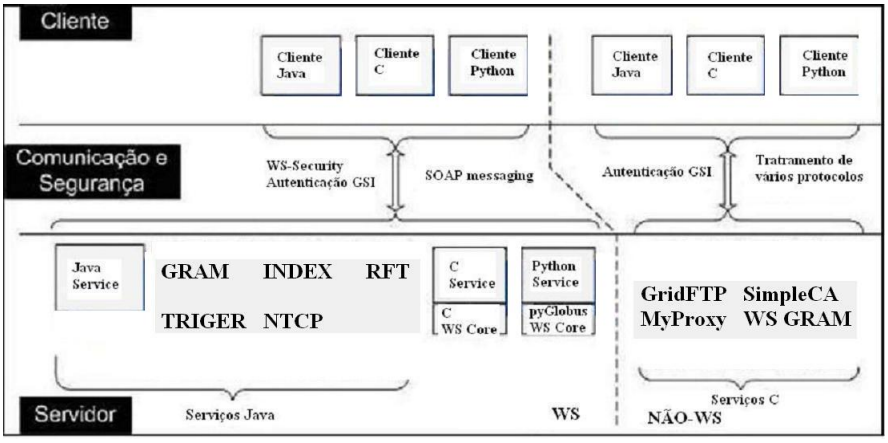

Figura 2.2: Componentes que formam o GT4 [26].

A Figura 2.2 está dividida por linhas horizontais, que representam o domínio cliente, segurança e comunicação e domínio servidor, respectivamente. No domínio cliente estão representadas aplicações desenvolvidas em diferentes linguagens.

Na parte referente ao domínio servidor estão representados os serviços do GT4 que compõem a base de sua arquitetura. Os serviços do GT4, WS ou não-WS, podem ser habilitados ou desabilitados conforme a urgência de cada ambiente.

Para alguns serviços adicionais do GT4, como o *GridFTP* as interfaces WS ainda não estão disponíveis, mas existe uma tendência que estes serviços sejam incorporados à tecnologia *Web Service*.

Web Service Resource Framework

O *Web Service*, geralmente, é usado nos casos em que as aplicações não necessitam armazenar o estado do recurso anterior à última requisição. Mas para aplicações em *Grid*, onde o conhecimento destes estados dos recursos é necessário, a tecnologia *Web Service*, sozinha, não consegue atender esse requisito [41].

Então, a *Organization for the Advancement of Structured Information Standards (OASIS)* , definiu o padrão *Web Service Resource Framework* (WSRF), um *framework* aberto à utilização, para a modelagem e ingresso aos estados dos recursos usando *Web Service* [37].

O WSRF não foi agregado a tecnologia *Web Service* , tendo sido criado para ser usado em conjunto com os serviços *web*, adicionando essa nova funcionalidade.

Foram criados separadamente por motivo de simplicidade. O WSRF foi colocado em uma entidade chamada *resource*, a qual guarda todas as informações referentes aos estados que uma certa tarefa impõe ao recurso que a está executando.

Para haver a identificação dos recursos há o uso de uma chave única, para que todas as vezes que for necessária uma interação com o recurso é só instruir o *Web Service* a empregar um recurso em particular através de sua chave.

Um exemplo simples da utilização do WSRF guardando os estados das requisições está representado na Figura 2.3 onde um cliente envia números a um *Web Service* para que esse faça a soma com os outros números previamente enviados e retorna ao cliente o valor, até o momento, acumulado.

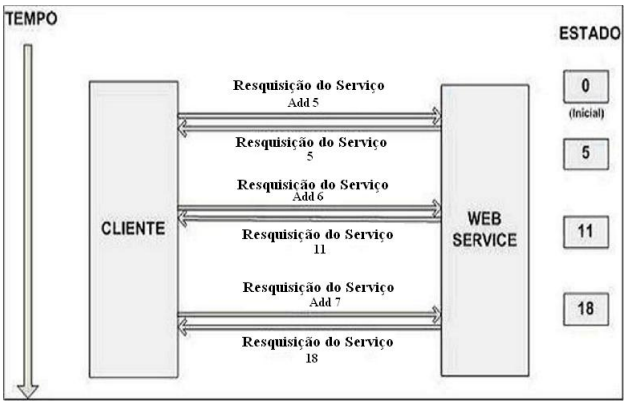

Figura 2.3: Funcionamento do WSRF [41].

A Figura 2.3 mostra que os estados anteriores são guardados no servidor para posterior soma e envio correto dos valores ao cliente.

Inicialmente, o cliente requisita para o servidor uma operação Add 5, para adicionar o número 5 ao estado inicial, que no caso, era zero. Quando é feita a segunda requisição, o cliente solicita a operação

Add 6, nesse caso o valor não será mais somado ao valor inicial, e sim ao valor acumulado, que é no caso 5. O WSRF possibilitará o acúmulo dos resultados das operações sucessivamente.

Os *resources* (recursos) são entidades lógicas que devem ser identificáveis. Possuem um conjunto de propriedades que podem ser expressas em um conjunto de informações XML e pode ter um tempo de vida (*lifecycle*). Juntamente com o *Web service*, os recursos formam o *WS-Resource* que são referenciados pela especificação *WS-Addressing*. O endereço de um *WS-Resource* é denominado de *endpoint* reference [42].

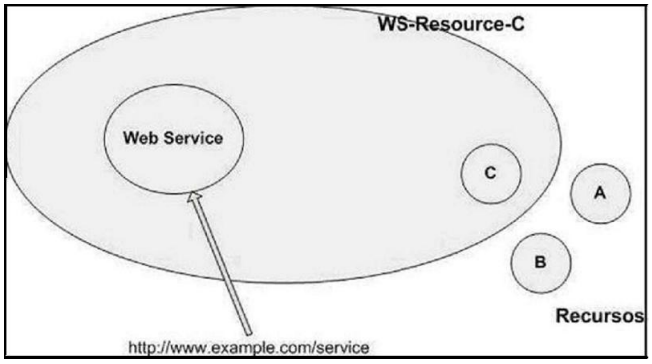

Figura 2.4: Representação de um *resource* [42].

A Figura 2.4, mostra o *WR-Resource-C* que é formado pelos serviços *web* localizados em:

http://www.example.com/service e pelo resource "C", sendo referenciado pelo esquema XML da Figura 2.5.

```
<wsa:EndpointReference>
        <wsa:Address>
                http://www.example.com/service?res=C
        </wsa:Address>
        ...
</wsa:EndpointReference>
```

Figura 2.5: Arquivo XML representando o *resource C* [42].

Containers

O código-fonte do GT4 inclui implementações para as principais aplicações do *Grid*. Algumas dessas aplicações são normas para todas as aplicações em *Grid* (ex. *WSRF e WS-Notification*) e outros são exclusivos do *Globus Toolkit* (GRAM e RTF).

Para a construção dos containers (caixas no domínio cliente da Figura 2.6) podemos combinar componentes fornecidos pelo *Globus Toolkit* com outros componentes presentes em WS com o protocolo SOAP. O *Simple Object Access Protocol* (SOAP) é um protocolo para troca de mensagens entre com-

putadores baseado no padrão XML [24]. Esse protocolo pretende, de forma simples, possibilitar que aplicações clientes se comuniquem com serviços remotos e invoquem métodos de forma remota. A proposta inicial do SOAP é a invocação de procedimentos remotos transportados via HTTP, mas o SOAP é flexível e pode ser utilizado com outros protocolos de transporte.

De uma forma geral, todos os *containers* do GT podem:

- Implementar o SOAP através de HTTP como protocolo de troca de mensagens, e níveis de transporte e segurança WS para a comunicação entre os membros do *Grid*;
- Implementar as funcionalidades de endereçamento *WS-Notification e WSRF*;
- Definir recursos *WS-WSRF* com propriedades para acessar informações sobre serviços em cada *container*. Além das características dos serviços descritas acima, especificamente, os *containers* GT4 podem prover serviços de hospedagem nos quais as interfaces dos clientes são definidos de acordo com as especificações básicas *Web Service* [26];
- Podem prover serviços de hospedagem que utilizam WSRF e mecanismos relacionados. Se forem instalados os *softwares* apropriados, podem também prover serviços avançados de hospedagem no *container* Java do GT4. Aproveitando GRAM, MDS e RFT.

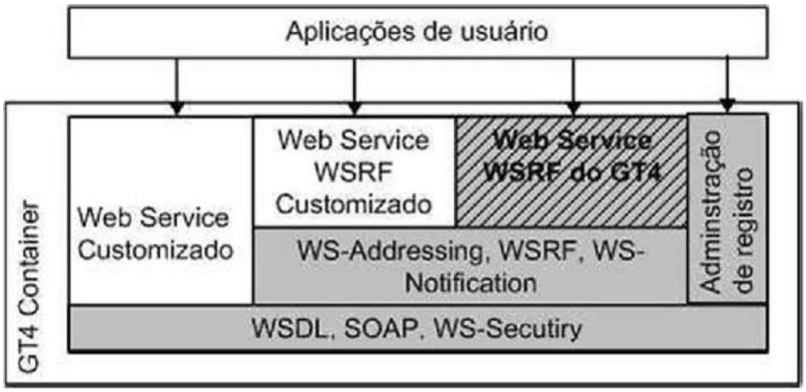

Figura 2.6: *Container* do GT4 [26].

A Figura 2.6, representa *containers* GT4, onde os *containers* possuem diferentes serviços de hospedagem, variando desde serviços básicos (ex. WS Customizado) até serviços opcionais avançados (ex. *GT4 WSRF-Web Service*).

2.2.3 Gerenciamento de Execução

O gerenciamento de execução consiste na inicialização, monitoramento, gerenciamento, escalonamento ou coordenação dos *jobs* submetidos, remotamente, aos recursos computacionais [17]. Os *Jobs*

são tarefas computacionais que realizam operações de entrada/saída. À medida que são executados, os *jobs* irão afetando os estados e sistemas de arquivos (*file system*) dos recursos computacionais. Na prática, estes *jobs*, antes ou depois de sua execução, podem precisar saber como está a situação dos dados nos recursos computacionais. Alguns podem interagir entre si acessando os resultados de outros *jobs* que ainda estão em execução [18].

O monitoramento consiste na requisição e notificação dos consumidores (*subscriber*) das informações de *status* como as mudanças de estado de um *job* [18].

Grid Resource Allocation Management

O *Globus Toolkit* 4 provê um conjunto de funcionalidades de gerenciamento em um único *software* nomeado *Grid Resource Allocation Management* (GRAM). Mas nessa versão do GT, é mais conhecido por WS GRAM, pois provê um conjunto de *web* services para realizar tais funções e para diferenciar dos GRAM *pré-web services* que existiam nas versões anteriores.

Tipos de *jobs* que devem utilizar o GRAM

O GRAM é utilizado para tratar um grande conjunto de *jobs*, onde programas arbitrários, operações consistentes, monitoramento de estados, gerenciamento de credenciais e os estágios dos arquivos são necessários [18].

Quando estes requisitos não se tornam necessários a uma aplicação, o GRAM não é uma boa escolha.

O GRAM apresenta vários componentes:

- Componentes de protocolo;
- Componentes de *software*;
- *Softwares* de segurança;
- *Softwares* de gerenciamento de *jobs*;
- *Softwares* de gerenciamento de dados;
- *Softwares* de gerenciamento de tarefas.

O GRAM não é monolítico e dentre os seus vários módulos pode-se customizar a sua instalação de acordo com o que é necessário entre os vários requisitos apresentados.

Funcionamento do WS GRAM

O WS GRAM combina serviços de gerenciamento de *jobs*, e adaptadores locais com outros componentes do GT4.0 para fazer a execução de *jobs* junto com a coordenação dos estágios de transferência de arquivos.

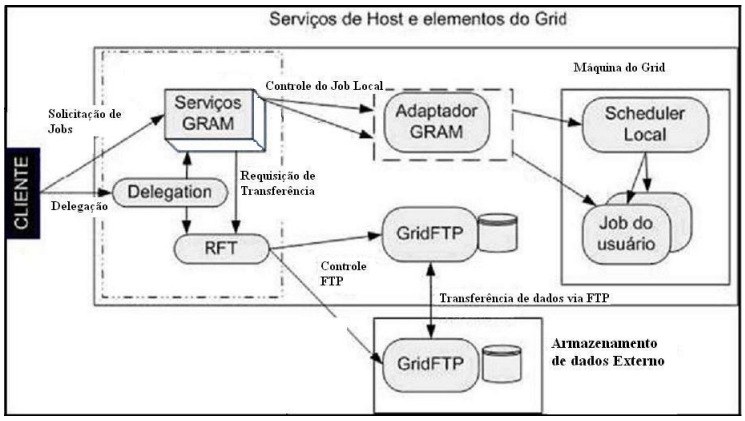

Figura 2.7: Visão macro do funcionamento do WS GRAM [19].

As trocas de mensagens entre os componentes da Figura 2.7 são feitas com o uso de *web services* que foram projetados para serem hospedados no WSRF do *Globus Toolkit*. Para cada submissão de um novo *job*, é criado um serviço denominado *ManagedJob*, no computador *host*. Se obtiver sucesso, na submissão, a operação de criação retorna um *EndPoint Reference* (EPR) para o *ManagedJob*. Com esse EPR, o cliente que o iniciou pode monitorar o *status* ou pode terminar a execução do *job*.

Cada elemento computacional, que é acessado por um escalonador local, é visto como um recurso distinto dos outros, sendo qualificado como um servidor *ManagedJobFactory* que disponibiliza uma interface para a criação dos serviços *ManagedJob* que serão gerenciados pelo escalonador local.

Principais passos da execução

A principal etapa do WS GRAM é a de criação de um recurso *ManagedJob* que é criado a partir de uma invocação do método *ManagedJobFactory* [19].

Um cliente que deseja começar uma tarefa no *Grid* deverá fazer uma chamada a esse método. Quando criado, o recurso terá um tempo de vida que será monitorado pelo GRAM até seu término. A destruição de um *ManagedJob* é um outro passo que poderá ocorrer com uma requisição explícita do cliente ou pela configuração do tempo de escalonamento. Cada administrador do sistema define um tempo *default* máximo que um *ManagedJob* pode ficar em execução até ser automaticamente extinto.

2.2.4 Monitoramento e Descoberta

Os mecanismos de monitoramento e descoberta interagem com recursos e serviços obtendo, distribuindo, indexando, arquivando e algumas vezes processando informações sobre a atual configuração e estado que estes recursos e serviços se encontram [26].

No *Globus Toolkit* 4, esse mecanismo é denominado *Monitoring and Discovering System* (MDS) sendo que nessa versão do GT é conhecido por MDS ou WSMDS pois é formado por um conjunto de *web services*.

O MDS permite fazer o monitoramento dos recursos que fazem parte das organizações virtuais com o uso de interfaces que permitem:

1. A requisição de informações de serviços WSRF;
2. A execução de programas para a aquisição de dados;
3. A interface com outros sistemas de monitoração.

2.2.5 Segurança

Em qualquer ambiente de execução distribuído é necessária a existência de bons mecanismos de segurança. O GT4 possui um grande número de ferramentas que são responsáveis por estabelecer identidade (autenticar) de usuários ou recursos, proteger o processo de troca de mensagens, definir o que cada membro do ambiente pode acessar e gerenciar credenciais de cada grupo de membros [27].

O GT4 provê formas diferentes (WS e pré-WS) de autenticar e autorizar suas transações, sendo que ambas as formas seguem o mesmo padrão. Basicamente identificam-se entidades persistentes, usuários e servidores, para suportar temporariamente as funções e privilégios de outras entidades, respectivamente. Esse processo é denominado de autenticação mútua, ou seja, após o primeiro elemento ser autenticado ele pode fornecer o certificado a um segundo elemento ainda não autenticado.

O conceito central do *Grid Security Infrastructure* (GSI) é o certificado. Todos os usuários e serviços do *Grid* no *Globus Toolkit* 4 são identificados via certificado. Os certificados GSI seguem o padrão X.509, que é um padrão internacional de certificado estabelecido pela *internet Engineering Task Force* (IETF). O aproveitamento do formato X.509 possibilita que os certificados possam ser utilizados por qualquer aplicação que obedeça esse padrão.

Nos certificados X.509, o validador é uma Autoridade de Certificação (CA). Como a GSI utiliza como base a emissão de certificados é importante verificar a confiabilidade da CA.

Todos os certificados X.509 possuem os seguintes dados [33]:

- Número da versão do X.509: indica o formato de acordo com a versão, afetando que informações estarão incluídas no certificado;
- Chave pública do possuidor do certificado: além da chave é fornecido um algoritmo de identificação especificando o sistema de criptografia e parâmetros pertinentes;
- Número de série do certificado: o membro do ambiente do *Grid* (usuário ou serviço) emite um número diferenciando para cada certificado que é emitido;
- Período de validade do certificado: indica o tempo de duração do certificado, corresponde à data de expiração do mesmo;
- Nome único do emissor do certificado: nome único dentro do ambiente, geralmente corresponde ao nome de uma CA;
- Assinatura digital do emissor: assinatura que utiliza chave privada da entidade que emitiu o certificado;
- Identificação do algoritmo de assinatura: fornece o algoritmo usado pela CA para assinar o certificado.

O GT 4 tem sido amplamente utilizado para desenvolvimento de ambientes *Grid*, um exemplo é o SINAPAD [13] brasileiro que foi totalmente estruturado utilizando o *Globus Toolkit*.

2.2.6 Legion

A ferramenta Legion teve seu desenvolvimento iniciado em 1993. É um sistema baseado em objetos, onde tudo, desde a capacidade de execução, memória e espaço de armazenamento é considerado como um objeto [25]. Sua idéia tem como base criar uma arquitetura de *Grid* provendo uma única máquina virtual para as aplicações do usuário, deixando algumas características do *Grid*, como escalabilidade, tolerância a falhas e segurança totalmente transparentes para seus usuários finais.

O Legion pode ser definido como uma camada intermediária que conecta redes, estações de trabalho e outros recursos em um sistema abrangendo diferentes sistemas operacionais e localizações físicas [30].

O Legion utiliza um serviço global de identificação, onde os objetos podem ser identificados por um mecanismo de nomeação em três níveis [39]. No nível mais alto, são identificados por uma cadeia de caracteres legíveis chamada de *context names*. No nível intermediário é usado um identificador binário chamado Legion *Object Identifier* (LOID), que possui uma chave pública RSA associada no momento da criação do objeto. Como o LOID não possui informações suficientes para que os objetos possam se

comunicar através da rede, é utilizado o terceiro método de nomeação, o *Legion Object Address* (LOA). É um endereço físico que contém as informações necessárias para a comunicação dos objetos, como o endereço IP, número de porta, etc.

Utilizando-se a combinação de objetos persistentes e o serviço global de identificação descrito acima, tem-se a noção de estar utilizando um sistema de arquivos tradicional no Legion. Isto simplifica a manipulação de arquivos para os programadores, que utilizam os conceitos de caminhos, diretórios e arquivos globalmente acessíveis, não importando a localização física dos mesmos. Além disso, podem ainda adicionar em seus programas características de tolerância à falhas, usando mecanismo para desfazer (*rollback*) ou recuperar dados.

2.2.7 Diferenças do *Globus* para o Legion

Tanto o *Globus* quanto o Legion são sistemas distribuídos de alto desempenho e que tem como objetivo encontrar meios de tornar a computação mais rápida, eficiente, fácil e acessível para usuários e programadores. Deste modo, existem grandes semelhanças entre eles, e a principal diferença está nos princípios de arquitetura e desenho. O *Globus* pode ser caracterizado como uma soma de serviços enquanto que o Legion possui uma arquitetura mais integrada [30].

Globus utiliza-se de conjuntos de componentes já existentes que são agrupados em um *Toolkit*. Por exemplo, existem serviços para agendamento, autenticação e submissão de *jobs*. Quando um novo serviço é projetado, o desenvolvedor é responsável por criar uma interface entre este novo serviço e os já existentes. Não existe uma arquitetura comum, onde todas as partes possam se comunicar de maneira simples. Enquanto isso, o Legion define uma estrutura com a qual todos as partes se comunicam. Assim, quando um nova parte é desenvolvida ela pode utilizar essa camada comum e sua integração com o restante do sistema é mais simples.

Essa diferença pode se tornar mais importante no futuro, pois quanto mais componentes vão sendo desenvolvidos para os *Grid*, maior é a complexidade, principalmente utilizando-se a arquitetura do Globus.

2.3 Portais para ambientes de *Grid*

O *middleware Globus Toolkit* quando totalmente implantado em um ambiente *Grid* pode trazer novas funcionalidades ao ambiente. Funcionalidades que podem trazer benefícios tanto para a parte de usabilidade como para a parte de segurança. Este trabalho apresenta duas funcionalidades, desenvolvimento de um portal *Grid* e a integração do *Globus Toolkit* com o gerenciador de *jobs PBS(Torque)*.

Para que um ambiente de *Grid* seja bem aproveitado e de fácil utilização é necessário utilizar ferra-

mentas que facilitem o acesso dos usuários aos recursos. Sem tais ferramentas, um usuário que não tiver bom conhecimento para usar várias linhas de comando, criar *scripts* para submeter e administrar tarefas, verificar recursos disponíveis, fazer transferências de arquivos, entre outras atividades, terá muita dificuldade de utilizar um ambiente de *Grid*. Os Portais *Grid* surgiram com a finalidade de facilitar este acesso aos recursos. Este tipo de portal disponibiliza os serviços oferecidos por um ambiente de *Grid* via *web*, facilitando tanto o acesso dos usuários quanto o trabalho dos administradores do ambiente de *Grid*, além de deixar mais claras todas as funcionalidades disponíveis.

Hoje existem várias empresas que desenvolvem ferramentas que criam aplicações para *web* vinculadas a *Grid*, porém com licenças muito caras. Por esse motivo, a comunidade de *software* livre tem desenvolvido ferramentas para criação de portais *Grid*.

Alguns exemplos são o *Grid Portal Development Kit* (GPDK) [29], *Legion Grid Portal* [30], *Grid Portal Toolkit* (GridPort) [43] e o *Gridsphere* [6], que são abordados a seguir.

2.3.1 *Grid Portal Development Kit*

A idéia do GPDK é desenvolver componentes comuns que podem ser usados por desenvolvedores para construir um portal que pode autenticar usuários de forma segura e ajudá-los a tomar decisões precisas no momento de agendarem *jobs*. Para isso, permite que os usuários tenham acesso a informações sobre os recursos alocados no *Grid*. Além disso, os usuários também podem visualizar e monitorar os jobs criados e seus resultados [29].

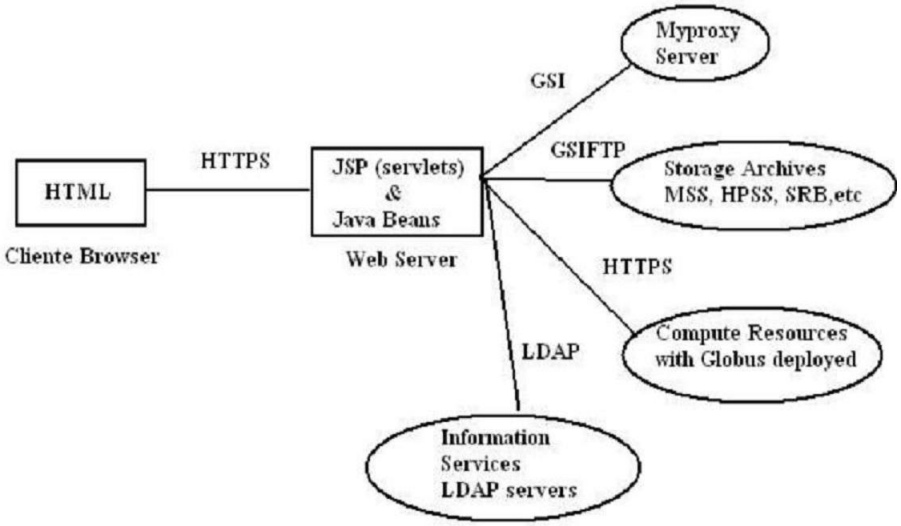

Figura 2.8: Arquitetura do GPDK [29].

A Figura 2.8 demonstra a arquitetura do GPDK. Ela foi desenvolvida sobre o modelo de três camadas, onde um cliente aproveitando um navegador *web* acessa, de modo seguro, via o protocolo

HTTPS [7], um servidor *web*. Este servidor *web* pode acessar vários serviços do *Grid* utilizando a ferramenta *Globus*. O *Globus*, por sua vez, permite a criação segura de novos *jobs*, pesquisa por informações de *hardware e software*, e mecanismos de segurança usando GSI.

O componente *Myproxy* é responsável por manter credenciais e permissões de usuários que podem ser utilizadas em qualquer parte do portal. Deste modo, os usuários podem usar o portal para garantir acesso a recursos remotos a partir de qualquer lugar, não necessitando que seus certificados ou chaves privadas estejam armazenados na máquina cliente onde o navegador *web* está sendo executado.

Como demonstra a Figura 2.8, o GPDK tem seu funcionamento baseado em tecnologias Java, principalmente *Java Server Pages* (JSP) e *Java Beans*. Utiliza como servidor *web* o aplicativo Tomcat, que é um servidor *open source* de grande utilização no mercado e desenvolvido pela *Apache Foundation*. Os *Java Beans* que compõem o GPDK são, em grande parte, derivados de funcionalidade contidas no *Globus Java Commodity Grid* (CoG) toolkit. O CoG provê uma série de APIs em Java para os recursos do *Globus Toolkit*. Utiliza, por exemplo, as bibliotecas Java SSL para fornecer acesso a GSI, implementa transferência de arquivos (*GSIFTP*) e pesquisa em servidores LDAP através da API *Java Naming and Directory Interface* (JNDI). Através dos beans do GPDK torna-se mais fácil a utilização do CoG para desenvolver os portais.

2.3.2 *Legion Grid Portal*

A ferramenta *Legion Grid Portal* tem por finalidade o desenvolvimento de uma interface amigável com a qual os usuários podem interagir com o ambiente de *Grid*. Para isso, utiliza-se de normas e softwares existentes para facilitar o ingresso à infra-estrutura do *Grid*. O *Legion Grid Portal* pode ser definido como uma arquitetura para integrar tecnologias existentes sobre uma interface comum [35].

A Figura 2.9 demonstra a arquitetura do *Legion Grid Portal*

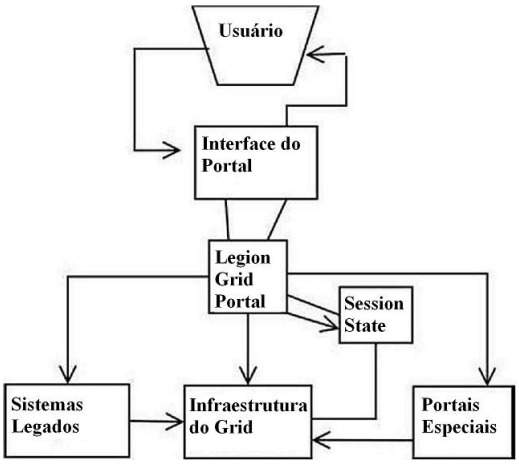

Figura 2.9: Arquitetura do *Legion Grid Portal* [35].

Como demonstra a Figura 2.9, a arquitetura do *Legion Grid Portal* é baseada em camadas. As camadas mais altas são representadas pelos usuários e pela interface do portal.

O componente central, o *Legion Grid Portal*, é um *script* CGI14 desenvolvido em Perl, e tem como finalidade processar as requisições feitas pelos usuários através da interface. Durante a execução, o portal gera informações de sessão e *cache*, que são usadas para fins de autenticação e melhoria na performance das execuções.

2.3.3 *Grid Portal Toolkit* (GridPort)

O *Grid Portal Toolkit* (GridPort) é uma coleção de tecnologias utilizadas para auxiliar no desenvolvimento de portais científicos em *Grid* computacionais, portais de usuários, interfaces de aplicativos e portais educacionais [43].

As páginas e os dados são gerados por módulos e bibliotecas, desenvolvidas na linguagem de programação Perl [11], armazenadas no servidor, enquanto que no lado do cliente, são utilizadas páginas em HTML. Deste modo, podem ser visualizadas em qualquer navegador *web*.

O *GridPort* é separado em duas partes principais, que são um conjunto de páginas HTML e *scripts* CGI, que formam os componentes da interface e uma camada de *software* que faz a ligação com as tecnologias de *Grid* . É uma coleção de *scripts* desenvolvidos em Perl, que o portal pode empregar para interagir com o *Globus* ou outras ferramentas de *Grid* [43].

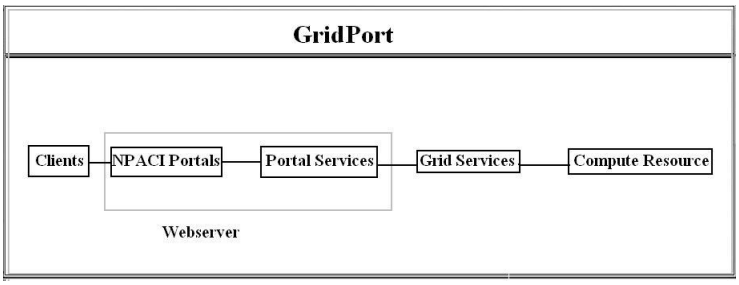

Figura 2.10: Arquitetura do *GridPort* [43].

No diagrama da Figura 2.10, as diferentes partes do sistema são demonstradas na forma de camadas, como mostra a Tabela 2.1. Cada uma destas camadas representa uma parte lógica do portal, onde dados e requisições de serviços fluem e tratam alguns aspectos específicos ou funções do portal [43].

2.3.4 *Gridsphere*

Gridsphere é um projeto *open source* desenvolvido como parte do projeto *GridLab*, fundado pela *European Commission*, e tem como principal finalidade a criação de um *framework* para o desenvolvimento de portais de *Grid* computacionais baseado no conceito de *portlets*. *Gridsphere* permite que

Tabela 2.1: Arquitetura do *GridPort*

Camada	Descrição
Clients:	São navegadores *web*, PDAs ou outros dispositivos móveis;
NPACI Portals:	São portais de aplicação que podem estar armazenados em outros servidores, mas que usam os mesmos componentes do *GridPort*;
Portal Services:	Além de realizar a mediação entre as requisições dos clientes e os serviços do *Grid* , outros serviços são prestados, como gerenciamento de sessões,coleções de arquivos e monitoramento;
Grid Services e Compute Resources:	Nestas camadas estão os componentes responsáveis pela interação com as ferramentas do *Grid* e os recursos alocados a este.

desenvolvedores rapidamente desenvolvam e empacotem aplicações *web portlets* de terceiros de modo que possam ser executadas e administradas dentro do *Gridsphere portlet container* [6].

Portlets são definidos como componentes visuais que podem ser assimilados dentro de páginas de um portal *web* [6]. Eles provêem pequenos aplicativos que podem mostrar conteúdo informacional ou prover ingresso a outros serviços. Com o aproveitamento de *portlets* em um portal, os usuários podem customizar a aparência e funcionalidades que deseja acessar. Isso devido ao fato de que todo portlet possui características que permitem ao usuário configurá-lo.

A partir da versão 2.0 do *Gridsphere*, dois modelos de desenvolvimento de portlets são suportados. O primeiro modelo do *Gridsphere* é baseado na API utilizada no IBM *Websphere* v4.1 e superiores. Isso garante que portlets desenvolvidos e executados no *software* da IBM facilmente possam ser portados para o *Gridsphere*. O segundo modelo é uma implementação do padrão *Java JSR 168 Portlet API16*, o que garante compatibilidade com aplicativos desenvolvidos por vários líderes de mercado, que também seguem esta especificação, tais como IBM [8], Sun [14], Oracle [9], entre outras.

Os componentes apresentados na Figura 2.11 são os componentes principais do *Gridsphere* que são combinados com o *container servlet* do servidor de aplicações *web*.

Quando o cliente, usando o navegador *web (browser)*, envia uma requisição ao portal o *Gridsphere servlet* é invocado, o qual por sua vez requisita ao mecanismo de *layout* do portal que converte os dados formando a saída para o usuário. Ambos, o *servlet* e o mecanismo responsável pela conversão do *layout*, fazem uso dos *core portlets* e *services do Gridsphere*. Os principais *portlets* fornecidos para atender requisitos básicos estão descritos na Tabela 2.2.

No quesito segurança, o *Gridsphere* usa o conceito de controle de acesso baseado em papéis (*role based access control*). Grupos podem ser criados e são definidos por um nome, descrição e uma coleção de *portlets* associados a ele. Como cada usuário pode pertencer a mais de um grupo, estes podem ter

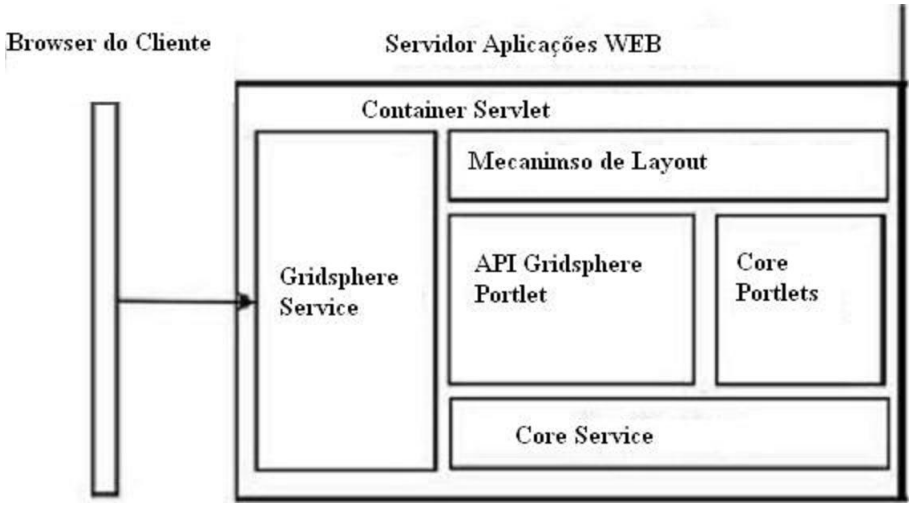

Figura 2.11: Arquitetura do *Gridsphere* [36].

acesso a várias aplicações *web* baseadas em *portlets*. Um usuário desempenha um papel (*role*) em cada um dos grupos a que pertence. Os quatro papéis suportados são *guest* (visitante), *user* (usuário), *admin* (administrador) e *super* (super usuário).

2.3.5 *Torque(Open PBS)*

O *Globus Toolkit* 4 pode ser integrado com diferentes tipos de gerenciadores de *jobs*, como o *PBS*, o *SGE* [15], o *CONDOR* [1] entre outros. Neste trabalho abordaremos somente a integração prática do *Globus Toolkit* com o PBS.

O PBS *(Portable Batch System)* [10] é um subsistema em rede para: submeter, monitorar e controlar a carga de processamento de *jobs*. O PBS possibilita que os usuários aloquem nós do *cluster* para obterem acesso exclusivo a estes e assim rodem suas aplicações sem a interferência de outras aplicações. Esta funcionalidade é interessante no momento em que fragmenta logicamente o *cluster* em diversos *clusters* menores, visto que o efeito de uma alocação é a requisição de um subconjunto dos nodos do *cluster* para serem utilizados na execução de uma aplicação. Para controlar o uso excessivo do *cluster* por parte de apenas um usuário, prejudicando os demais interessados, o PBS possibilita a implantação de uma política de uso para o *cluster*.

A política de uso é um conjunto de regras que arbitra a quantidade máxima de nós e de horas que uma reserva (alocação) pode ter. A operação básica do PBS é implementada como uma fila FIFO (*First In First Out*), onde os *jobs* (processos dos usuários) são executados de acordo com sua ordem de submissão. Sendo assim, processos submetidos à fila ficam a espera de sua vez pela execução, estando sujeitos a duração dos processos que já constam na fila. O PBS também oferece mecanismos para visualização e interação com a fila de execução através de gráficos demonstrando os *jobs* já submetidos

e uma interface simplificada para alocação de nós. O PBS fornece também estatísticas a respeito da taxa total de utilização do *cluster*, além destes dados por usuário.

O servidor de tarefas é o foco central para o PBS. Todos os comandos se comunicam com o usuário através de uma rede.

O executor do trabalho é o *daemon* que coloca realmente o trabalho na execução. Este *daemon*, pbs-mom, é chamado informalmente de *MOM* porque é a mãe de todos os trabalhos a executar. O *MOM* coloca um trabalho na execução quando recebe uma cópia do trabalho de um servidor. O *MOM* cria uma sessão nova que seja tão idêntica a uma sessão do início de uma sessão do usuário quanto é possível.

O *daemon* do *scheduler* de trabalho, pbs-sched, implementa a política de controle de quando cada trabalho será executado e em quais recursos. O *scheduler* comunica-se com os vários MONS para questionar o estado de sistema de recursos e com o servidor que vai avaliar os trabalhos a executar. A interface do servidor é com o mesmo API que usada pelos comandos do cliente.

Como curiosidade o PBS pode ser chamado também de *Torque*, pois o projeto *OpenPBS* foi descontinuado, e então um grupo de pesquisadores resolveu continuar o projeto do OpenPBS dando-lhe o nome de *Torque*, com isso tanto o PBS quanto o *Torque* se referem ao mesmo gerenciador de *jobs*.

2.4 Considerações Finais

Este capítulo mostrou os principais conceitos em computação em *Grid*, seus benefícios juntamente com os componentes utilizados. Mostrou algumas ferramentas que auxiliam no desenvolvimento de ambientes *Grid*. Por fim foi dado uma introdução sobre portais *Grids*, quais são sua importância e como desenvolvê-lo.

Tabela 2.2: Arquitetura do *Gridsphere* [36]

Camada	Descrição
Login:	Permite que os usuários acessem o portal usando um nome e uma senha;
Logout:	Realiza o processo de saída do portal por parte do usuário;
Locale Selection:	O usuário pode escolher dentre algumas configurações de localização, alterando a língua e demais opções relacionadas. As configurações existentes na versão atual são: Inglês, Francês, Alemão, Italiano, Húngaro, Tcheco e Polonês;
Account Request:	Fornece a interface com a qual um novo usuário pode requisitar a criação de uma conta no portal, podendo escolher também configurações avançadas, como, por exemplo, em qual grupo deseja se filiar;
Account Management:	Permite que usuários com permissão de administração possam controlar as características das contas dos demais usuários;
User Management:	Fornece aos administradores a capacidade de aprovar ou recusar as requisições de criação de novas contas ou a afiliação destas nos grupos;
User Profile:	Os usuários podem definir as características de seus perfis, tais como nome completo, e-mail, etc. Também permite que os usuários possam escolher os portlets com os quais irão interagi;
Layout Configuration:	Os usuários podem configurar o modo como são visualizados os portlets, como sua localização e aparência;
Portlet Subscription:	Fornece mecanismos com os quais o usuário pode adicionar ou remover portlets de seu espaço de trabalho;
Local File Manager:	Com este recurso os usuários têm acesso a um sistema de arquivos "virtual"onde podem editar, enviar e copiar arquivos para o portal;
Notepad:	Usuários podem gerenciar notas, podendo criar, excluir e realizar buscas;
Text Messaging:	Fornece o recurso de mensagens instantâneas de texto facilitando a comunicação entre os usuários do portal.

Capítulo 3

Metodologia

No início do trabalho foi elaborada uma pesquisa sobre ambientes *Grid*, para entender os detalhes do seu funcionamento e levantar em quais sistemas operacionais um ambiente *Grid* poderia ser elaborado. Apartir deste estudos optou-se por escolher o sistema operacional *Fedora Core Linux*.

Devido à sua grande utilização iniciou-se o estudo em cima do *middleware Globus Toolkit* e levantou-se seus principais pré-requisitos de instalação e configuração que encontram-se no Apêndice A.

Para efetuar as instalações necessárias foram utilizadas duas maquinas Pentium 4 Duo Core, nos quais foi instalado e configurado o ambiente *Grid*. Após toda a instalação e configuração foram realizados alguns testes, executando aplicações para poder verificar a eficiência e o poder do ambiente.

Toda metodologia do trabalho, com as principais instalações, as principais configurações das ferramentas, estão demostradas nos diagramas abaixo, para dar uma visão geral do que será efetuado ao longo deste capítulo. Os diagramas se referem respectivamente as instalações do *middleware Globus Toolkit 4*(Figura 3.1), instalação do *Gridsphere*(Figura 3.2), do *Torque (OpenPBS)*(Figura 3.3) e suas integrações(Figura 3.4).

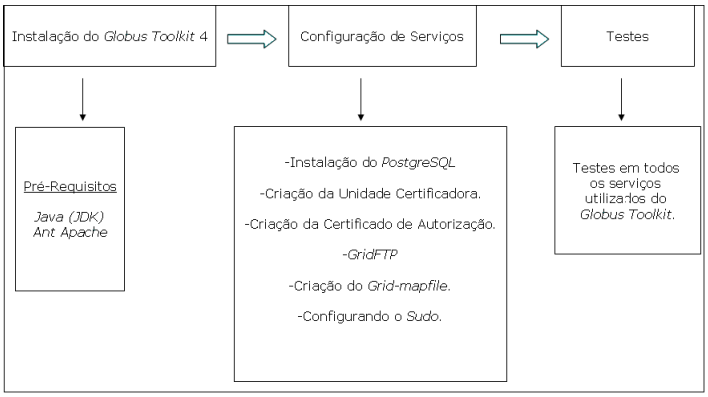

Figura 3.1: Instalação do *Globus Toolkit 4*

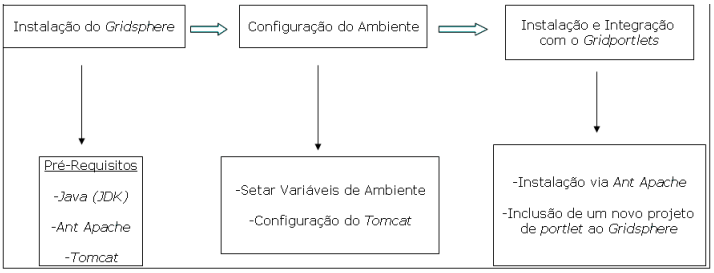

Figura 3.2: Instalação do *Gridsphere*

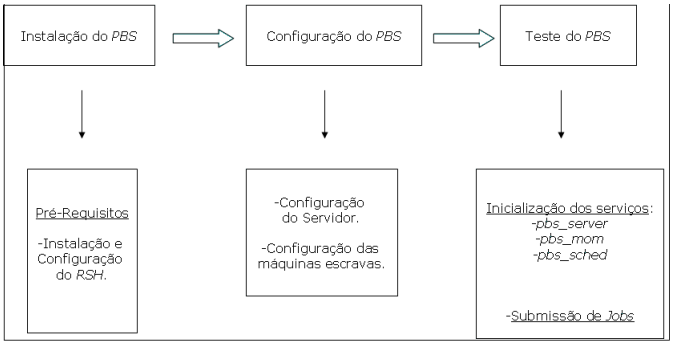

Figura 3.3: Instalação do *Torque (OpenPBS)*

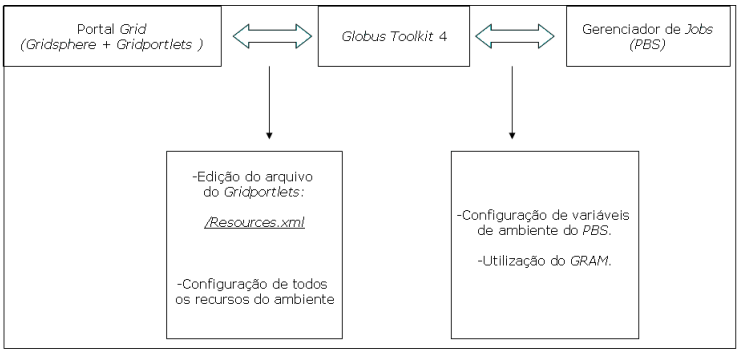

Figura 3.4: Integrações das ferramentas utilizadas

3.1 Instalação do *Globus Toolkit* 4

Esta Seção apresenta o passo a passo de como foi instalado e configurado o *Globus Toolkit* 4. O *Globus Toolkit* 4 pode ser instalado em diferentes tipos de sistemas operacionais, tais como *SuSe Linux 9/8* [16], *Red Hat Linux 9* [12], *Fedora Core Linux* [4] e *Debian 3.1* [2]. A instalação realizada neste trabalho foi para o sistema operacional *Fedora Core 6*. Vale ressaltar que a instalação para outras versões do *Globus Toolkit*, como a versão anterior 3.0 ou posteriores à versão 4, podem conter diferenças significativas de instalação.

3.1.1 Pré-Requisitos para a Instalação do *Globus Toolkit* 4

Antes de iniciar a instalação e configuração do *Globus Toolkit* e de seus serviços é preciso instalar e configurar alguns programas que são importantes para o pleno funcionamento do *middleware*. A lista destes *softwares* está na Tabela 3.1. Todos os pré-requisitos juntamente com seu tutorial de instalação e configuração está disponível no Apêndice A.

Tabela 3.1: Lista de softwares necessários para a instalação do *Globus Toolkit*

Nome do Software	Versão Recomendada
Java JDK (IBM / Sun / BEA)	1.4.2 ou posterior
Apache Ant	1.5.1 ou posterior
gcc	3.2.1 e 2.95.x
GNU tar	-
GNU sed	-
zlib	1.1.4 ou posterior
GNU Make	-
sudo	-
PostgreSQL	7.1 ou posterior

3.1.2 Instalação e Configuração do *Globus Toolkit* 4

Há dois tipos de instalação do *Globus Toolkit* 4: *source* ou binário. Neste trabalho foi realizada a instalação *source* que leva cerca de 83 minutos de instalação.

Preliminares

Primeiramente é necessário criar o diretório onde o *Globus Toolkit* será instalado. Em seguida, é preciso alterar o grupo deste diretório para globus.

```
[root@nodeB ~]# mkdir -p /opt/globus-4.0.1
[root@nodeB ~]# chown globus.globus /opt/globus-4.0.1
[root@nodeB ~]# ls -alh /opt | grep globus
drwxr-xr-x 2 globus globus 4.0K Feb 20 12:05 globus-4.0.1
```

Os passos seguintes da configuração devem ser realizados com o usuário 'globus':

```
[root@nodeB opt]# su - globus
[globus@nodeB ~]$ cd
```

O Download do *Globus Toolkit* 4 em modo source pode ser baixado através do link:

http://www.globus.org/ftppub/gt4/4.0/4.0.1/installers/src/gt4.0.1-all-source-installer.tar.bz2

Após o download verificar qual o grupo ele pertence:

```
[globus@nodeB ~]$ ls -l gt4.0.1-all-source-installer.tar.bz2
-rw-rw-r-- 1 globus globus 88609887 Aug 5 2005 gt4.0.1-all-source-installer.tar.bz2
```

O md5 checksum do arquivo é:

```
[globus@nodeB ~]$ md5sum gt4.0.1-all-source-installer.tar.bz2
398076812364f53d694194320836b8ad gt4.0.1-all-source-installer.tar.bz2
```

Instalando

O arquivo gt4.0.1-all-source-installer.tar.bz2 deve ser descompactado:

```
[globus@nodeB ~]$ tar -jxf gt4.0.1-all-source-installer.tar.bz2
```

Em seguida, deve-se setar as variáveis do java e do ANT para o usuário globus:

```
[globus@nodeB ~]$ export JAVA_HOME=/opt/jdk1.5.0_06
[globus@nodeB ~]$ export PATH=$JAVA_HOME/bin:$PATH
[globus@nodeB ~]$ which java
/opt/jdk1.5.0_06/bin/java
[globus@nodeB ~]$ which javac
/opt/jdk1.5.0_06/bin/javac
[globus@nodeB ~]$ export ANT_HOME=/opt/apache-ant-1.6.5
[globus@nodeB ~]$ export PATH=$ANT_HOME/bin:$PATH
[globus@nodeB ~]$ which ant
/opt/apache-ant-1.6.5/bin/ant
```

Ao entrar no diretório de instalação do *Globus Toolkit*, setar a variável de ambiente:

GLOBUS_LOCATION para o diretório criado anteriormente /opt/globus-4.0.1 e configurar a instalação para o diretório da variável GLOBUS_LOCATION

```
[globus@nodeB ~]$ cd gt4.0.1-all-source-installer

[globus@nodeB gt4.0.1-all-source-installer]$ export GLOBUS_LOCATION=/opt/globus-4.0.1

[globus@nodeB gt4.0.1-all-source-installer]$ ./configure --prefix=$GLOBUS_LOCATION
checking build system type... i686-pc-linux-gnu
checking for javac... /opt/jdk1.5.0_06/bin/javac
checking for ant... /opt/apache-ant-1.6.5/bin/ant
configure: creating ./config.status
config.status: creating Makefile
To build the toolkit simply run 'make':
[globus@nodeB gt4.0.1-all-source-installer]$ make
```

Após o comando *make* o arquivo de instalação estará compilando as classes para que possa ser efetuada a instalação do *Globus Toolkit* 4. Como a distribuição utilizada é *source* ela irá demorar aproximadamente 80 minutos. Após compilar todas as classes, pode-se prosseguir a instalação.

```
[globus@nodeB gt4.0.1-all-source-installer]$ make install
```

Criando Certificado de Autorização

Após a instalação é necessário criar uma unidade certificadora. Note que só é necessário criar um certificado autoridade em uma máquina do *Grid*, ou seja, independentemente do tamanho do *Grid* só é preciso de uma unidade certificadora.

A configuração do ambiente para o usuário globus é feita com os comandos:

```
export GLOBUS_LOCATION=/opt/globus-4.0.1
source $GLOBUS_LOCATION/etc/globus-user-env.sh
```

O comando *setup-simple-ca*, inicia o processo de instalação da unidade certificadora. É solicitado por duas vezes, uma senha ou frase. Também é necessário adicionar alguns dados ao longo da instalação como o nome da organização, o domínio em qual se encontra a unidade certificadora e um e-mail. Como a instalação é um pouco grande será colocado passo a passo a instalação no Apêndice A.

```
[globus@nodeB gt4.0.1-all-source-installer]$
$GLOBUS_LOCATION/setup/globus/setup-simple-ca
```

O comando *setup-gsi* faz com que a CA que acabou de ser criada torne-se o padrão de certificado da unidade certificadora.

```
[globus@nodeB ~]
$ /opt/globus-4.0.1/setup/globus_simple_ca_f1f2d5e6_setup/setup-gsi
 -default -nonroot

setup-gsi: Configuring GSI security
Making trusted certs directory: /opt/globus-4.0.1/share/certificates/
mkdir /opt/globus-4.0.1/share/certificates/
Installing /opt/globus-4.0.1/share/certificates//grid-security.conf.f1f2d5e6...
Running grid-security-config...
Installing Globus CA certificate into trusted CA certificate directory...
Installing Globus CA signing policy into trusted CA certificate directory...
setup-gsi: Complete
```

Obtendo Certificado Para a Entidade Certificadora

Para fazer um pedido de certificação para a entidade certificadora. Pois num ambiente de *Grid*, é necessário ter uma unidade certificadora, porém ela pode gerar pedidos de certificação onde outros computadores possam fazer o papel da unidade certificadora, podendo assim gerar e assinar credenciais para os demais usuários do certificador. Isso faz com que diminua o trabalho e o controle das credenciais em somente um computador, podendo descentralizar esse controle.

Após o pedido ser assinado, o certificado deve ser instalado no local.

Para requisitar um certificado, primeiramente, deve-se setar as variáveis:

```
[root@nodeB opt]# export GLOBUS_LOCATION=/opt/globus-4.0.1
[root@nodeB opt]# source /opt/globus-4.0.1/etc/globus-user-env.sh
```

O comando "grid-cert-request"cria os pedidos de certificação. O arquivo hostcert_request.pem e o arquivo hostkey_resquest.pem, para sua criação devemos informar o domínio da máquina e o diretório onde serão copiados.

```
[root@nodeB opt]# grid-cert-request -host nodeb.ufla.br -dir $GLOBUS_LOCATION/etc
```

Arquivo hostcert_request.pem:

```
[root@nodeB opt]# ls -alh $GLOBUS_LOCATION/etc/hostcert_request.pem
-rw-r--r-- 1 root root 1.3K Feb 22 10:39 /opt/globus-4.0.1/etc/hostcert_request.pem
```

Arquivo hostkey:

```
[root@nodeB opt]# ls -alh $GLOBUS_LOCATION/etc/hostkey.pem
-r-------- 1 root root 887 Feb 22 10:39 /opt/globus-4.0.1/etc/hostkey.pem
```

Agora que o certificado foi solicitado, o pedido deve ser assinado pela CA (unidade certificadora). Para assinar o pedido deve-se entrar como usuário 'globus' e configurar o ambiente novamente:

```
[root@nodeB opt]# su - globus
[globus@nodeB ~]$ export GLOBUS_LOCATION=/opt/globus-4.0.1
[globus@nodeB ~]$ source $GLOBUS_LOCATION/etc/globus-user-env.sh
```

O pedido de certificado é assinado usando o comando "grid-ca-sinal". Quando solicitado deve-se digitar a senha para o certificado autoridade:

```
[globus@nodeB ~]$ grid-ca-sign -in $GLOBUS_LOCATION/etc/hostcert_request.pem -out
 $GLOBUS_LOCATION/etc/hostcert.pem
```

Nota: Se o arquivo hostcert.pem já existir, o comando *grid-ca-sign* pode ser executado com a opção -f (forçar) .

Para assinar o pedido é preciso digitar a senha para a CA-Key.O novo certificado assinado é armazenado em: /home/globo/.globus/simpleCA//newcerts/01.pem.Antes de iniciar demais serviços do globus, é preciso ter certeza de que as permissões e propriedade do arquivo estão corretos. Os arquivos devem ser de propriedade do usuário *root* com as permissões mostradss a seguir:

```
[root@nodeB opt]# chown root.root /opt/globus-4.0.1/etc/hostcert.pem

[root@nodeB opt]# chmod 644 /opt/globus-4.0.1/etc/hostcert.pem

[root@nodeB opt]# ls -alh /opt/globus-4.0.1/etc/host*.pem
-rw-r--r-- 1 root root 2.5K Feb 22 10:45 /opt/globus-4.0.1/etc/hostcert.pem
-r-------- 1 root root 887 Feb 22 10:39 /opt/globus-4.0.1/etc/hostkey.pem
```

Copiando Para o Container

O certificado criado é de propriedade do root e será utilizado por serviços como o *globus-gridftp-servidor*. Na maioria das vezes os outros serviços e os conteiners não são executados como *root*, mas como usuário 'globus'. Por isso, é preciso fazer uma cópia do certificado para que o usuário tenha acesso ao *Globus Toolkit*. Esta cópia deve ser realizada como usuário *root*:

```
[root@nodeB opt]# cp /opt/globus-4.0.1/etc/hostcert.pem
/opt/globus-4.0.1/etc/containercert.pem

[root@nodeB opt]# chown globus.globus /opt/globus-4.0.1/etc/containercert.pem

[root@nodeB opt]# ls -alh /opt/globus-4.0.1/etc/containercert.pem
-rw-r--r-- 1 globus globus 2.5K Feb 22 10:57 /opt/globus-4.0.1/etc/containercert.pem

[root@nodeB opt]# cp /opt/globus-4.0.1/etc/hostkey.pem
/opt/globus-4.0.1/etc/containerkey.pem

[root@nodeB opt]# chown globus.globus /opt/globus-4.0.1/etc/containerkey.pem

[root@nodeB opt]# ls -alh /opt/globus-4.0.1/etc/containerkey.pem
-r-------- 1 globus globus 887 Feb 22 10:58 /opt/globus-4.0.1/etc/containerkey.pem
```

O arquivo GLOBUS_LOCATION/etc/globus_wsrf_core/global_security_descriptor.xml deve ser configurado para que o *Globus Toolkit* ache o diretório dos certificados. Configure de forma correta este arquivo com os respectivos diretórios dos certificados.

```
[globus@nodeB opt]$ cat /opt/globus 4.0.1/etc/globus_wsrf_core
/global_security_descriptor.xml
<?xml version="1.0" encoding="UTF-8"?>
<securityConfig xmlns="http://www.globus.org">
<credential>
<key-file value="/opt/globus-4.0.1/etc/containerkey.pem"/>
<cert-file value="/opt/globus-4.0.1/etc/containercert.pem"/>
</credential>
<gridmap value="/opt/globus-4.0.1/etc/grid-mapfile"/>
</securityConfig>
```

Criando o grid-mapfile

O arquivo *grid-mapfile* deverá ser criado e tem como finalidade fazer um mapa do *Grid*, ou seja fazer listagem de todos usuários que tenham acesso aos serviços do *Grid* com seus domínios, respectivamente.

```
[globus@nodeB ~]$ touch $GLOBUS_LOCATION/etc/grid-mapfile
```

Configurando o Serviço RFT

O *Reliable File Transfer* (RFT) é um mecanismo para fazer persistência do estado das transferências em armazenamento confiável, sua interface é baseada em protocolos de *web services* (SOAP), funciona como um escalonador de jobs para transferências de arquivos.

RFT Permite consultar o estado das transferências, e evita que o usuário tenha que manter a conexão durante transferências demoradas.

Seu servidor está disponível através de um serviço no *container* do *Globus Toolkit*, utilizando bancos de dados de terceiros, como o *PostgreSQL*.

O RFT deve estar vinculado ao PostgreSQL. Para verificar este vínculo, o comando a seguir deve ser executado:

```
[root@nodeB opt]# ps auwwwwx|grep postmaster
postgres 26088 0.0 0.3 19476 3172 pts/0 S
/usr/bin/postmaster -D /opt/pgsql/data -o -i
```

Como usuário 'postgres' deve-se criar um usuário 'globus' no banco de dados (*createuser*). As opções -A e -d , fazem com que o usuário globus tenha as permissões corretas. É solicitado uma senha para o usuário. Recomenda-se uma senha sem espaços ou caracteres estranhos e que seja de fácil memorização:

```
[root@nodeB opt]# su - postgres
```

```
-bash-3.00$ createuser -A -d -P globus
Enter password for new user:
Enter it again:
CREATE USER
```

O arquivo pg-hba.conf (/opt/pgsql/data/pg_hba.conf) dá permissões ao usuário 'globus', pela adição de algumas linhas que permitirão que o usuário 'globus' consiga se conectar com o banco de dados.

É necessário adicionar a seguinte linha ao final do arquivo:

host rftDatabase "globus192.168.31.40"255.255.255.255 md5

O próximo passo é criar as tabelas que o usuário 'globus' necessita. Isto é feito como usuário 'globus':

```
[root@nodeB ~]# su - globus
[globus@nodeB ~]$ export GLOBUS_LOCATION=/opt/globus-4.0.1
[globus@nodeB ~]$ source $GLOBUS_LOCATION/etc/globus-user-env.sh
```

Como usuário globus, a execução do comando '*createdb*' cria o banco de dados RFT:

```
[globus@nodeB ~]$ createdb rftDatabase
CREATE DATABASE
```

A inicialização do banco de dados é feita com o comando:

```
[globus@nodeB ~]$ psql -d rftDatabase -f $GLOBUS_LOCATION/share/globus_wsrf_rft
/rft_schema.sql
psql:/opt/globus-4.0.1/share/globus_wsrf_rft/rft_schema.sql:6: NOTICE: CREATE TABLE
/ PRIMARY
 KEY will create implicit index "requestid_pkey" for table "requestid"
CREATE TABLE

psql:/opt/globus-4.0.1/share/globus_wsrf_rft/rft_schema.sql:11: NOTICE: CREATE TABLE
/ PRIMARY
 KEY will create implicit index "transferid_pkey" for table "transferid"
CREATE TABLE

psql:/opt/globus-4.0.1/share/globus_wsrf_rft/rft_schema.sql:30: NOTICE: CREATE TABLE
/ PRIMARY
 KEY will create implicit index "request_pkey" for table "request"
CREATE TABLE

psql:/opt/globus-4.0.1/share/globus_wsrf_rft/rft_schema.sql:65: NOTICE: CREATE TABLE
/ PRIMARY
```

```
 KEY will create implicit index "transfer_pkey" for table "transfer"

CREATE TABLE
CREATE TABLE
CREATE TABLE
CREATE INDEX
```

Configurando o *Sudo*

Muitos serviços do *Globus Toolkit* precisam usar o comando *sudo*, a fim de executar processos com diferentes tipos de usuários.

Como usuário *root* o arquivo /etc/sudoers deve ser editado e adicionadas as duas linhas seguintes:

Nota: O seguinte texto aparece em várias linhas, mas cada entrada deve ser digitado como uma linha de texto

```
globus ALL=(nemesio) NOPASSWD: /opt/globus-4.0.1/libexec/globus-gridmap-and-execute
-g /opt/globus-4.0.1/etc/grid-mapfile /opt/globus-4.0.1/libexec
/globus-job-manager-script.pl *

and

globus ALL=(nemesio) NOPASSWD: /opt/globus-4.0.1/libexec/globus-gridmap-and-execute
-g /opt/globus-4.0.1/etc/grid-mapfile /opt/globus-4.0.1/libexec
/globus-gram-local-proxy-tool *
```

Essas duas linhas permitirá ao usuário 'globus' usar o *sudo* para executar comandos para o usuário 'nemesio' (usuário genérico). É possível usar qualquer usuário genérico em seu sistema, basta mudar o nome da conta nas linhas acima.

Iniciando o *Container*

Com a unidade certificadora e chaves disponíveis e com o RFT configurado o *Globus Container* pode ser iniciado, onde dará início a execução dos demais serviços do *Grid*. Como usuário globus, para certificar-se que as variáveis de ambiente do java estão no *PATH* são necessários os comandos:

```
[globus@nodeB ~]$ export JAVA_HOME=/opt/jdk1.5.0_06
[globus@nodeB ~]$ export PATH=$JAVA_HOME/bin:$PATH
[globus@nodeB ~]$ which java
/opt/jdk1.5.0_06/bin/java
```

É melhor iniciar o container com a memória padrão utilizada pela máquina virtual java. Isso é realizado através da definição de uma variável de ambiente:

```
[globus@nodeB ~]$ export GLOBUS_OPTIONS=-Xmx512M
```

O comando '*globus-start-container*' é usado para iniciar o *container* e todos os services do *Grid*:

```
[globus@nodeB ~]$ $GLOBUS_LOCATION/bin/globus-start-container
Starting SOAP server at: https://192.168.31.40:8443/wsrf/services/
With the following services:
[1]: https://192.168.31.40:8443/wsrf/services/TriggerFactoryService
[2]: https://192.168.31.40:8443/wsrf/services/DelegationTestService
[3]: https://192.168.31.40:8443/wsrf/services/SecureCounterService
[4]: https://192.168.31.40:8443/wsrf/services/IndexServiceEntry
[5]: https://192.168.31.40:8443/wsrf/services/DelegationService
[6]: https://192.168.31.40:8443/wsrf/services/InMemoryServiceGroupFactory
[7]: https://192.168.31.40:8443/wsrf/services/mds/test/execsource/IndexService
[8]: https://192.168.31.40:8443/wsrf/services/mds/test/subsource/IndexService
.
.
.
.
[48]: https://192.168.31.40:8443/wsrf/services/ContainerRegistryService
[49]: https://192.168.31.40:8443/wsrf/services/TestAuthzService
[50]: https://192.168.31.40:8443/wsrf/services/CASService
[51]: https://192.168.31.40:8443/wsrf/services/ManagedJobFactoryService
```

Iniciando o *Globus-Gridftp-Server*

O *globus-gridftp-server* é executado como *root*. Antes de iniciar o servidor GridFTP ao qual iniciará os serviços de tranferências de arquivos do *Grid*, é necessário definir a variável de ambiente do GRIDMAP para o grid-mapfile criado anteriormente:

```
[root@nodeB etc]# export GRIDMAP=/opt/globus-4.0.1/etc/grid-mapfile
```

O serviço é iniciado utilizando a porta padrão 2811.

```
[root@nodeB etc]# /opt/globus-4.0.1/sbin/globus-gridftp-server -p 2811
```

3.2 Integrando o *middleware Globus Toolkit* com o gerenciador de *jobs Torque (Open PBS)*

Antes de iniciar a integração do *middleware* com o gerenciador de *jobs* é necessário que o PBS esteja instalado e configurado corretamente no ambiente. Sua instalação pode ser verificada no Apêndice B.

Um dos serviços de submissão de *jobs* do *Globus Toolkit*, o GRAM WS, possui um gerenciador de *jobs* padrão, chamado *fork*, porém este gerenciado de *jobs* é muito limitado, o que pode trazer problemas na submissão de muitas tarefas mais complexas. Nesta seção será feita a integração do GRAM WS com um gerenciador de *jobs* mais eficientes, o *Torque (OpenPBS)*.

Antes da construção do PBS *jobmanager* é preciso ter certeza de que os caminhos dos serviços do *Torque (OpenPBS)* estão configurados corretamente. Segue os comandos:

```
[globus@nodeB ]$ export PATH=/opt/pbs/bin:$PATH
[globus@nodeB ]$ which qsub
/opt/pbs/bin/qsub
[globus@nodeB gt4.0.1-all-source-installer]$ which qstat
/opt/pbs/bin/qstat
[globus@nodeB gt4.0.1-all-source-installer]$ which pbsnodes
/opt/pbs/bin/pbsnodes
```

É preciso também definir a variável PBS_HOME para que o ambiente aponte para o diretório onde o servidor PBS escreve seus arquivos de *log*.

O GRAM WS PBS *jobmanager* está incluído no GT 4 fonte, assim é preciso ir ao diretório e fazer um *build* dele:

```
[globus@nodeB ~]$ cd gt4.0.1-all-source-installer
[globus@nodeB gt4.0.1-all-source-installer]$ make gt4-gram-pbs
[globus@nodeB gt4.0.1-all-source-installer]$ make install
running /opt/globus-4.0.1/setup/globus/setup-seg-pbs.pl..
[ Changing to /opt/globus-4.0.1/setup/globus ]
..Done
running /opt/globus-3.0.1/setup/globus/setup-globus-scheduler-provider-pbs..
[ Changing to /opt/globus-4.0.1/setup/globus ]
checking for pbsnodes... /opt/pbs/bin/pbsnodes
checking for qstat... /opt/pbs/bin/qstat
find-pbs-provider-tools: creating ./config.status
config.status: creating /opt/globus-4.0.1/libexec/globus-scheduler-provider-pbs
..Done
running /opt/globus-4.0.1/setup/globus/setup-gram-service-pbs..
[ Changing to /opt/globus-4.0.1/setup/globus ]
Running /opt/globus-4.0.1/setup/globus/setup-gram-service-pbs
..Done
```

O último passo é configurar o *jobmanager* para que ele saiba que está sendo usado o rsh:

```
[globus@nodeB globus]$ cd $GLOBUS_LOCATION/setup/globus
[globus@nodeB globus]$ ./setup-globus-job-manager-pbs --remote-shell=rsh
```

```
find-pbs-tools: WARNING: "Cannot locate mpiexec"
find-pbs-tools: WARNING: "Cannot locate mpirun"
checking for mpiexec... no
checking for mpirun... no
checking for qdel... /opt/pbs/bin/qdel
checking for qstat... /opt/pbs/bin/qstat
checking for qsub... /opt/pbs/bin/qsub
checking for rsh... /usr/kerberos/bin/rsh
find-pbs-tools: creating ./config.status
config.status: creating /opt/globus-4.0.1/lib/perl/Globus/GRAM/JobManager/pbs.pm
```

3.3 Instalando o *Gridsphere*

Gridsphere Portal Framework proporciona um portal *web open source* baseado em *portlets*. Permite rápido desenvolvimento de aplicações *web de portlets* que podem ser executadas e administradas dentro do próprio *Gridsphere*. É importante destacar que o *Gridsphere* é independente da aplicação *web*, por isso não contém nenhum suporte para tecnologia *Grid*. Esta funcionalidade será conseguida através do *portlet* chamado *Grid Portlet*.

Gridsphere apresenta as seguintes características: 1. Implementação de *Portlet API 100% compatible con JSR 168*; 2. Desenvolvimento de *portlets* usando *Java Server Faces* (JSF); 3. Suporte para rápido desenvolvimento e integração de aplicações com *portlets*; 4. XML para a configuração de apresentação do portal; 5. Suporte integrado para a administração de usuários.

Portlets são componentes Java incorporados a páginas *web* de um portal. São "mini aplicações"que podem mostrar informações e prover acesso a outros serviços. Os *portlets* são administrados pelo *portlets container*, processam requerimentos e geram conteúdo dinâmico como resposta.

Gridsphere suporta vários *portlets*, principalmente os modelos baseados na IBM *WebSphere Portlet API 4.1+*, e implementações de *Portlet API JSR 168*. Antes de iniciar a instalação do *Gridsphere* é necessário instalar 3 programas que são pré-requisitos, o JDK o ANT e o Tomcat. A instalação do ANT e do JDK está no Apêndice A, portanto segue uma breve idéia de instalação do tomcat.

O Tomcat pode ser obtido gratuitamente no site http://tomcat.apache.org/. Após sua instalação é necessário extrair seus arquivos para uma pasta, de preferência do usuário, e por fim setar uma variável de ambiente chamada CATALINA_HOME.

O *Gridsphere* pode ser obtido no endereço http://www.gridsphere.org/gridsphere/. Os arquivos devem ser extraídos para uma pasta de preferência do usuário, como exemplo opt/gridsphere/projects.

No diretório onde os arquivos foram descompactados deve-se executar o seguinte comando do *ant*, comando ao qual faz um *build* da aplicação do *gridsphere* :

```
[root@nodeB ~]$ ant install
```

Lembrando que para instalar tanto o tomcat como o *gridsphere* deve-se ter permissão de *root*.

Se tudo ocorrer bem após 5 minutos o *Gridsphere* já estará instalado. Antes de entrar no *Gridsphere*, o tomcat deve ser inicializado através do comando:

```
[root@nodeB ~]$ cd /usr/tomcat/bin
[root@nodeB ~]$ ./startup.sh
```

Utilizando um *browser* e digitando o endereço http://127.0.0.1:8080/gridsphere/, pode-se visualizar a tela início do *Gridsphere* (Figura 3.1).

Um detalhe importante para evitar erros é lembrar que o endereço http://127.0.0.1:8080/gridsphere/, é o ip default das máquinas. Este número pode ser diferente de máquina para máquina, dependendo da configuração de cada máquina.

Na Figura 3.5 pode-se observar a tela inicial do *Gridsphere*. É nela que os usuários e os administradores devem entrar com seus respectivos login e senha.

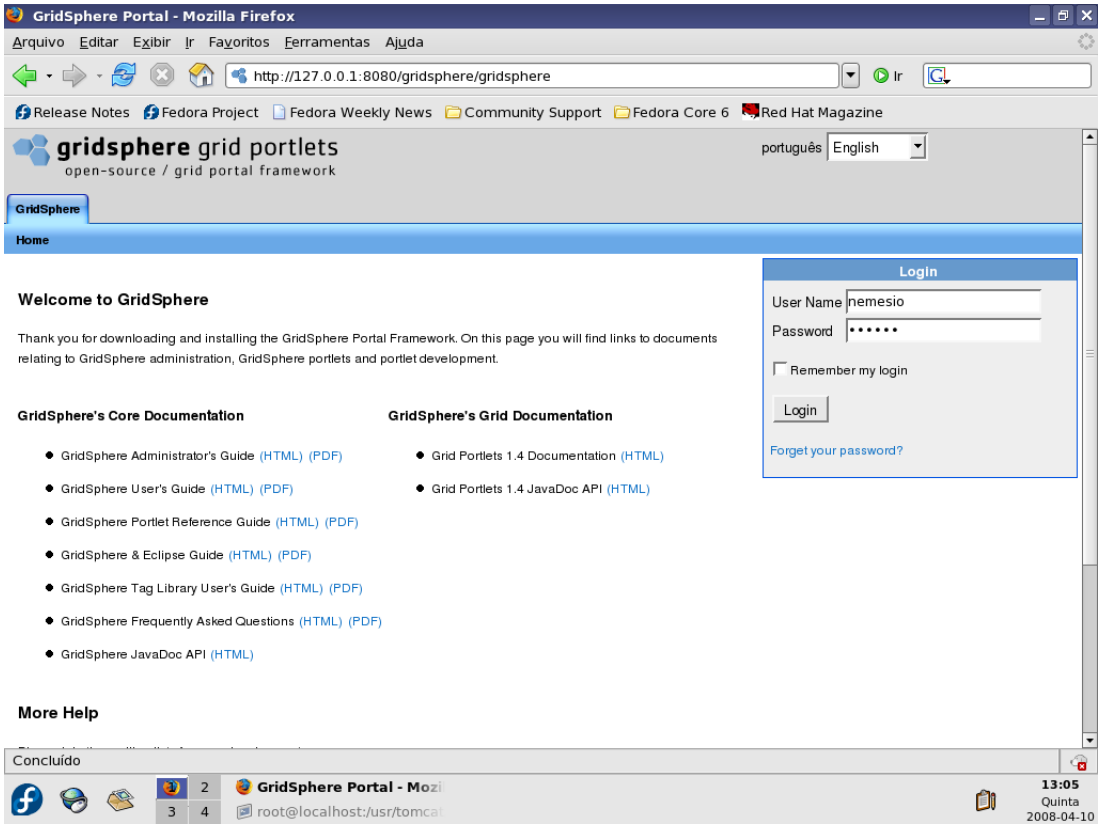

Figura 3.5: Tela inicial do *Gridsphere*.

3.4 Instalando o *Gridportlets*

Uma vez que o gridsphere esteja instalado corretamente. O *Gridportlets* pode ser obtido no site http://www.gridsphere.org/gridsphere/gridsphere/guest/download/.

A descompactação do arquivo é feita dentro do endereço /opt/gridsphere/projects, endereço ao qual foi descompactado o *Gridsphere*, e através do terminal executar o comando do *ant* que irá fazer o *build* do *Gridportlets*:

```
[root@nodeB ~]$ ant install
```

Após 5 minutos a instalação já terá sido concluída. Para colocar o *gridportlets* em funcionamento deve-se iniciar novamente o tomcat e entrar via *browser* com o endereço http://127.0.0.1:8080/gridsphere/. note na Figura 3.6 o aparecimento de uma nova aba com novas funcionalidades, este é o *gridportlets*.

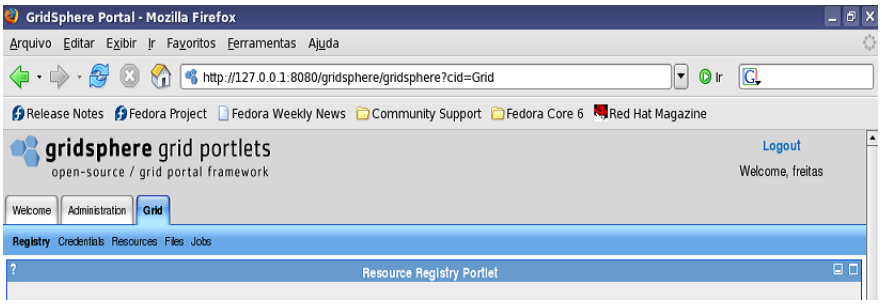

Figura 3.6: *Gridportlets* em funcionamento.

Uma observação importante: após a instalação ser feita deve-se editar o arquivo:

/webapp/WEB-INF/Resources.xml, de acordo com a Figura 3.7, para definir os recursos que estão sendo usados no portal *Grid*, como por exemplo quem será a máquina *front end* e as máquinas escravas. Essa edição pode ser feita pelo administrador do portal através de opção *Resource Registry Portlet*.

3.5 Considerações Finais

Ao final da instalação do *Gridsphere* e do *gridportlets*, juntamente com toda a configuração do *Globus Toolkit* 4, tem-se um ambiente *Grid* concluído. O passo seguinte é a realização de testes para verificar a eficiência do ambiente, o que veremos no próximo capítulo.

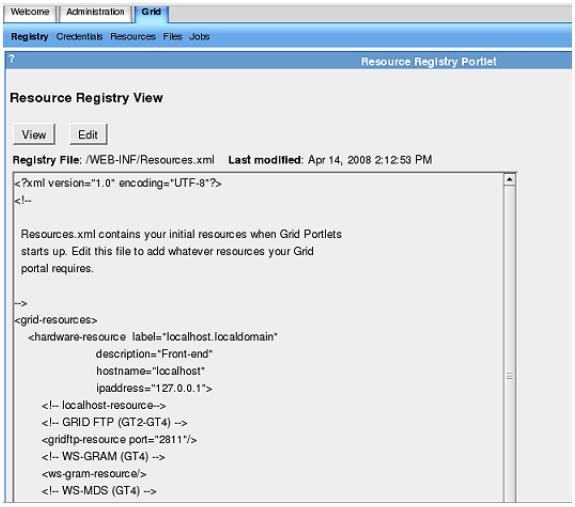

Figura 3.7: *Resources* do *gridportlets*.

Capítulo 4

Resultados

Este capítulo tem como objetivo discutir os principais testes feitos para verificar as funcionalidades dos serviços do ambiente *Grid*, os principais erros obtidos ao longo da implantação do ambiente e por fim algumas submissões para verificar a eficiência do portal juntamente com o ambiente *Grid*.

4.1 Testes

Nesta seção serão apresentados alguns testes dentro do ambiente *Grid*, com a finalidade de testar alguns serviços e alguns componentes do *Globus Toolkit*. Esses serviços são muito importantes para o pleno funcionamento do ambiente *Grid*.

Foram realizados seis testes: obtenção de certificado para usuário genérico, checagem da consistência do grid-mapfile, revalidação de uma credencial através da criação de um *proxy*, verificação da instalação e configuração do GridFT, verificação da instalação e configuração do RFT e verificação da instalação e configuração do WS GRAM.

4.1.1 Obtendo certificado para usuário genérico.

Para testar o funcionamento dos certificados digitais emitidos pela unidade certificadora (CA), foi emitido um certificado para um usuário genérico verificando assim a eficiência do serviço.

Foi usado no terminal um usuário genérico chamado "nemesio". Foi realizada a requisição de um novo certificado, com o comando abaixo:

```
[nemesio@hosta]$ grid-cert-request
Enter your name, e.g., Nemésio: grid user 1 (type grid user name)
A certificate request and private key is being created.
You will be asked to enter a PEM pass phrase.
This pass phrase is akin to your account password,and is used to protect your
key file.
```

```
If you forget your pass phrase, you will need to obtain a new certificate.
Generating a 1024 bit RSA private key
.....................................++++++
...++++++
writing new private key to '/home/auser1/.globus/userkey.pem'
Enter PEM pass phrase: (type pass phrase for grid user)
Verifying - Enter PEM pass phrase:(retype pass phrase for grid user)
...(unrelated information omitted)
```

Com a requisição do certificado feita, é necessário que a unidade certificadora (CA) assine, pois sem a assinatura da CA o certificado não tem validade nenhuma. Geralmente o certificado se encontra no diretório nemesio/.globus/usercert_request.pem, onde nemesio é o nome do usuário.

Geralmente em um ambiente *Grid*, todos os certificados requisitados são mandados via e-mail para a unidade certificadora.

O certificado gerado pelo usuário genérico agora deverá ser assinado, de acordo com o comando abaixo:

```
[globus@ca]$ grid-ca-sign -in usercert_request.pem -out usercert.pem
To sign the request
please enter the password for the CA key:
The new signed certificate is at:
/home/globus/.globus/simpleCA//newcerts/01.pem
```

Observe que quem assina o certificado é o usuário globus.

Após ter assinado o certificado, a CA deve mandar novamente o certificado para o usuário genérico e colocar esse novo certificado no diretório nemesio/.globus/.

Ao final deve-se testar se todos os atributos do certificado estão corretos como validade, nome, chaves, domínio e etc. Para fazer isso deve-se executar o comando:

```
[nemesio@hosta]$ grid-proxy-init -debug -verify
User Cert File: /home/nemesio/.globus/usercert.pem
User Key File: /home/nemesio/.globus/userkey.pem
Trusted CA Cert Dir: /etc/grid-security/certificates
Output File: /tmp/x509up_u511
Your identity:
/O=Grid/OU=GlobusTest/OU=simpleCA-ca.redbook.ibm.com/OU=redbook.ibm.com/CN=grid
user 1
Enter GRID pass phrase for this identity:
Creating proxy .........++++++++++++
..................++++++++++++
```

```
Done
Proxy Verify OK
Your proxy is valid until: Thu Jun 9 22:16:28 2008
```

4.1.2 Checando a consistência do *grid-mapfile*.

Globus Toolkit 4 requer um mapeamento entre todos os utilizadores do ambiente *Grid* e de seus domínios, afim de prover um mapa de todos seus usuários que tenham acesso ao ambiente. Para o usuário se adicionar no mapa do *Grid* basta executar os comandos abaixo:

```
[nemesio@hosta]$ grid-cert-info -subject -f /home/nemesio/.globus/usercert.pem
/O=Grid/OU=GlobusTest/OU=simpleCA-ca.nodb.ufla.br/OU=ufla.br/CN=grid
nemesio
```

Como usuário *root*, é permitido o acesso a este mapa e pode-se ver quantos e quais usuários estão mapeados no ambiente. Neste caso só há um usuário:

```
[root@hosta]# grid-mapfile-add-entry -dn \
"/O=Grid/OU=GlobusTest/OU=simpleCA-nodb.ufla.br/OU=ufla.br/CN=gri
d nemesio" -ln nemesio
Modifying /etc/grid-security/grid-mapfile ...
/etc/grid-security/grid-mapfile does not exist... Attempting to create
/etc/grid-security/grid-mapfile
New entry:
"/O=Grid/OU=GlobusTest/OU=simpleCA-nodb.ufla.br/OU=ufla.br/CN=gri
d nemesio" nemesio
```

Por fim, para verificar a consistência do mapfile, será submetido ao *Grid-mapfile-check-consistency*. Se não for recebida nenhuma resposta a partir deste comando, então significa que a grid-mapfile está coerente.

```
[root@hosta]# grid-mapfile-check-consistency
```

4.1.3 Revalidando uma credencial, criação de um *proxy*.

Toda credencial assinada e válida para ser utilizada por qualquer usuário no ambiente *Grid* tem um certo tempo para ser usada estabelecida pelo administrador da CA.

Quando esta validade chega ao fim, para não precisar fazer todo processo novamente, o *Globus Toolkit 4* tem um serviço de *proxy* para as credenciais que as recupera após sua validade, precisando somente estabelecer uma senha para a credencial. Isto é feito com o comando abaixo:

```
[nemesio@hosta]$ grid-proxy-init
Your identity:*****
/O=Grid/OU=GlobusTest/OU=simpleCA-nodb.ufla.br/OU=ufla.br/CN=grid
nemesio
Enter GRID pass phrase for this identity:
Creating proxy ..................................... Done
Your proxy is valid until: Tue Jun 14 21:41:25 2008
```

4.1.4 Verificando instalação e configuração do GridFTP.

O serviço do GridFTP é muito importante para transferências de arquivos em ambientes *Grid*, ele proporciona maior segurança e confiabilidade dos dados na hora da transferências. Para o teste foram realizados dois exemplos. No exemplos ocorre a transferência de um arquivo teste para um usuário genérico da mesma máquina (exemplo 1) e a transferência de um arquivo teste para um usuário genérico de uma máquina diferente (exemplo 2).

- Exemplo 1

```
[nemesio@hosta]$ echo "GridFTP Test" > /tmp/gridftptest.txt
[nemesio@hosta]$ globus-url-copy gsiftp://hosta/tmp/gridftptest.txt \
file:///tmp/gridftptest_copied.txt
[nemesio@hosta]$ cat /tmp/gridftptest_copied.txt
GridFTP Test
[nemesio@hosta]$ globus-url-copy file:///tmp/gridftptest_copied.txt \
gsiftp://hosta/tmp/gridftptest_copied2.txt
[auser1@hosta]$ cat /tmp/gridftptest_copied2.txt
GridFTP Test
```

- Exemplo 2

```
[nemesio@hosta]$ echo "ThirdParty GridFTP Test" > /tmp/thirdparty.txt
[nemesio@hosta]$ globus-url-copy gsiftp://hosta/tmp/thirdparty.txt \
gsiftp://hostb/tmp/thirdparty.txt
[nemesio@hosta]$ ssh usuario@hostb
usuario@hostb's password:
Last login: Thu May 9 19:36:31 2008 from nodb.ufla.br
[usuario@hostb]$ cat /tmp/thirdparty.txt
ThirdParty GridFTP Test
[usuario@hostb]$ ll /tmp/thirdparty.txt
-rw-r--r-- 1 usuario usuario 24 May 9 19:36 /tmp/thirdparty.txt
```

4.1.5 Verificando instalação e configuração do RFT.

Para verificar se o serviço RFT está instalado corretamente basta entrar no sistema como usuário globus e nele efetuar a inicialização do *container*. Se tudo estiver correto com o RFT, o *container* iniciará sem nenhum problema. A saída do comando é como se segue:

```
[globus@nodb]$ globus-start-container
Note: The postgres user is automatically generated during the
PostgreSQL package installation shown in Example 11-45 on page 180.
Chapter 11. {\itshape Globus Toolkit} 4 installation and configuration 183
Starting SOAP server at: https://192.168.1.103:8443/wsrf/services/
With the following services:
[1]: https://192.168.1.103:8443/wsrf/services/TriggerFactoryService
[2]: https://192.168.1.103:8443/wsrf/services/DelegationTestService
...(informações omitidas)
[51]: https://192.168.1.103:8443/wsrf/services/ManagedJobFactoryService
2008-02-03 21:46:49,805 INFO impl.DefaultIndexService
[Thread-9,processConfigFile:99] Reading default registration configuration from
file: /usr/local/globus-4.0.0/etc/globus_wsrf_mds_index/hierarchy.xml
```

4.1.6 Verificando instalação e configuração do WS GRAM.

O WS GRAM é um dos principais serviços do *Globus Toolkit*. É ele que permitirá que o *Globus Toolkit* submeta tarefas através de outros gerenciadores de *jobs*, como por exemplo o *Torque (OpenPBS)*. A seguir é apresentada a execução de uma tarefa teste para ver seu funcionamento. A única função desta tarefa é verificar se o serviço do WS GRAM está funcionando corretamente, retornando assim a palavra *True*.

```
[nemesio@hosta]$ globusrun-ws -submit -c /bin/true
Submitting job...Done.
Job ID: uuid:14add3c2-83c8-11dc-b007-49b27214df5e
Termination time: 03/27/2008 13:33 GMT
Current job state: Active
True
Current job state: CleanUp
Current job state: Done
Destroying job...Done.
```

Durante a implantação e a realização dos testes no ambiente *Grid* podem ocorrer alguns erros que estão relatados na seção 4.2.

4.2 Erros

Nesta seção são relatados os principais erros que ocorreram durante toda a implantação do *Globus Toolkit*. São erros comuns de difícil resolução, por isso esta seção tem como finalidade apresentar a resolução deste problemas.

- Erro1. A seguinte mensagem aparece durante a execução do comando *globus-start-container*

  Failed to start container: Failed to initialize 'ManagedJobFactoryService' service [Caused by: [SEC] Service credentials not configured and was not able to obtain container credentials.;

  Este erro pode estar acontecendo devido a criação errada de certificados. Além disso o erro mostra que o grid-mapfile não foi criado.

- Erro 2. A seguinte mensagem aparece durante a execução do comando *globus-start-container*

  Failed to start container: Container failed to initialize [Caused by: Address already in use]

  Esse erro ocorreu porque o container já está sendo executado no sistema. Para resolver este problema é necessário executar o comando *globus-stop-container* e reiniciar o *container*.

- Erro 3. A seguinte mensagem aparece durante a execução do comando *counter-create*.

  Error: ; nested exception is: GSSException: Defective credential detected [Caused by: Proxy file (/tmp/x509up_u511) not found.]

  O erro ocorreu devido a um acesso ao *container*, porém a credencial não estava válida, ou seja, ela pode ter expirado. Para ativar a credencial basta executar o comando *grid-proxy-init* para tornar a credencial válida e ter acesso ao *container*.

- Erro 4. Muita demora no tempo para transferir um pequeno arquivo de dados utilizando *globus-url-copy*

  O nome do servidor pode estar configurado incorretamente.

  Isto é verificado no arquivo: /etc/resolv.conf

- Erro 5. A seguinte mensagem aparece durante a execução do comando *globus-url-copy*

  globus_gsi_gssapi: Error with gss credential handle globus_credential: Valid credentials could not be found in any of the possible locations specified by the credential search order. Valid credentials could not be found in any of the possible locations specified by the credential search order.

  O erro ocorreu devido a um acesso ao *container*, porém a credencial não estava válida, ou seja, ela pode ter expirado.

Para ativar a credencial basta executar o comando *grid-proxy-init* para tornar a credencial válida e ter acesso ao *container*.

- Erro 6. A seguinte mensagem aparece durante a execução do comando *globus-start-container*.

  2008-05-04 21:41:12,135 ERROR service.ReliableFileTransferImpl [main,<init>:73] Unable to setup database driver with pooling. A connection error has occurred: FATAL: No pg_hba.conf entry for host (XXX.XXX.XXX.XXX), user globus, database rftDatabase

  Esta mensagem aparece porque *PostgreSQL* não está configurado corretamente. Para reparar o erro é preciso verificar no arquivo /var/lib/pgsql/data/pg_hba.conf se há uma entrada para o host e seu referente IP.

- Erro 7. A seguinte mensagem aparece durante a execução do comando *globus-start-container*.

  2008-04-13 16:10:55,374 ERROR service.ReliableFileTransferImpl [main,<init>:73] Unable to setup database driver with pooling.Connection refused. Check that the hostname and port are correct and that the postmaster is accepting TCP/IP connections.

  Esta mensagem aparece porque o *container* não pode conectar-se ao *PostgreSQL*. Para verificar se o *PostgreSQL* está configurado corretamente pode-se comparar com a instalação do *PostgreSQL* que está no Apêndice A. Se a instalação estiver incorreta deve-se instalar o *PostgreSQL* novamente.

4.3 Submissões de Tarefas

Nesta seção é apresentada a submissão de algumas tarefas ao ambientes *Grid* para testar sua eficiência e capacidade. Foram executadas 4 tarefas simples tanto pelo WS GRAM juntamente com o PBS quanto pelo portal *Grid*.

4.3.1 Submissão /bin/ls

Esta tarefa tem a finalidade de efetuar um ”ls” no diretório corrente do usuário. Primeiramente foi executado via WS GRAM e depois via portal *Grid*.

Antes de iniciar, vale lembrar que o *container* do *Globus Toolkit* deve ser iniciado de acordo com a metodologia do capítulo 3.

Com um terminal de um usuário genérico, foi efetuado o comando, ao qual é constituído por uma chamada ao *globusrun-ws* que tem a finalidade de gerenciar a submissão da tarefa, uma *flag -submit* que é a opção para submissão, outra *flag* -Ft ao qual faz referência a um outro gerenciador de *jobs (PBS)*,

não sendo utilizado o gerenciador de *jobs* padrão do *Globus Toolkit*, e por fim o endereço da máquina e o comando a ser executado:

```
[nemesio@nodb ~]\$ globusrun-ws -submit -F
https://nodb.ufla.br:8443/wsrf//ManagedJobFactoryService -Ft PBS -c /bin/ls
Submitting job...Done.
Job ID: uuid:547f0fd6-a4b6-11da-ac33-0011d8b1eb22
Termination time: 04/16/2008 21:49 GMT
Current job state: Pending
Current job state: Active
.....Saída.............
sg246778.pdf
torque-2.0.0p11.tar.gz
torque-2.2.1.tar.gz
trash.desktop
UPQuarterlyReport20080331e.pdf
XML-Parser-2.36
XML-Parser-2.36.tar.gz
.......................
Current job state: CleanUp
Current job state: Done
Destroying job...Done.
```

Após ter obtido o resultado via terminal com o WS GRAM, foi efetuado o mesmo exemplo utilizando o portal *Grid*. Primeiramente é necessário logar no portal. Para se logar basta seguir o exemplo da Figura 3.1. Para poder submeter uma tarefa através do portal existe a necessidade de possuir uma credencial para que o ambiente fique mais seguro. Para isso é preciso entrar com a credencial, criada na seção 3.1.2, para ativar a credencial já existente. A Figura 4.1 ilustra este passo:

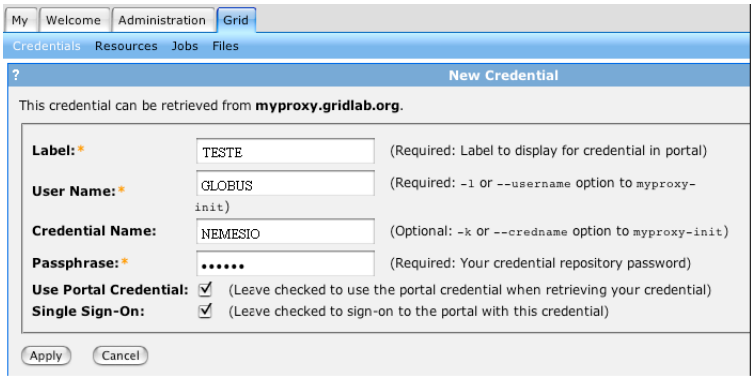

Figura 4.1: Ativação de credenciais via portal *Grid*.

Pode-se verificar na Figura 4.1, que em nenhum momento é preciso entrar no terminal e executar

algum comando do *Globus Toolkit* para ativar a credencial existente, o último passo a executar é entrar com os dados, quem tem o trabalho de ativar a credencial é o próprio portal.

Após ter entrado com a credencial, existe a possibilidade de ver todos os recursos do ambiente, todas as máquinas que estão ativas. A Figura 4.2 apresenta as máquinas existentes no ambiente *Grid*:

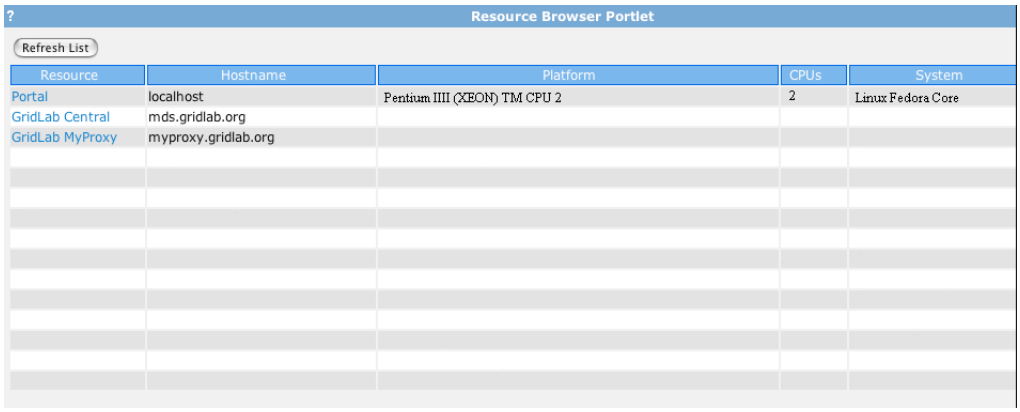

Figura 4.2: Listagem das máquinas ativas do ambiente.

Com a credencial e os recursos ativos pode-se iniciar a submissão da primeira tarefa, esta tarefa tem a finalidade de fazer um *"ls"*, ou seja, listar tudo que existe no diretório do usuário. Para isso temos que preencher alguns campos na tela do portal, como a descrição, ou título (*Description*) e o comando de execução /bin/ls (*Executable*). Existem outros campos para serem preenchidos no portal, porém está tarefa por ser simples não tem a necessidade de preenchimento. Tudo isso pode ser verificado na Figura 4.3:

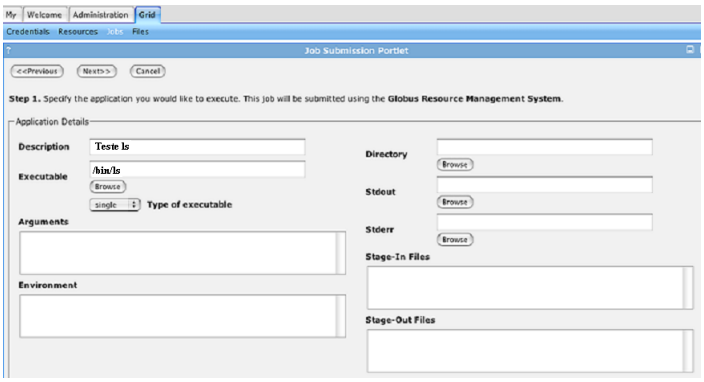

Figura 4.3: Início da submissão do Teste ls.

Como esta tarefa é simples não é preciso fazer nenhum upload de arquivo, basta colocar o comando na opção *executable*.

No segundo passo deve-se escolher em qual máquina o *job* será executado. Como exemplo foi uti-

lizado uma máquina apenas, e podemos escolher também qual o gerenciador de *jobs* que será utilizado, neste caso temos o *fork*, gerenciador *default* do *Globus Toolkit* e o *Torque (OpenPBS)*. (Figura 4.4).

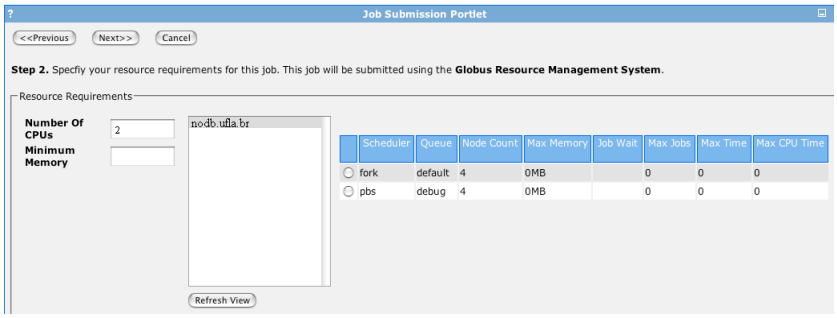

Figura 4.4: Escolha da máquina a qual ira ser executada a tarefa ls.

Após colocar os dados da tarefa, o próximo passo é confirmar e executar (Figura 4.5):

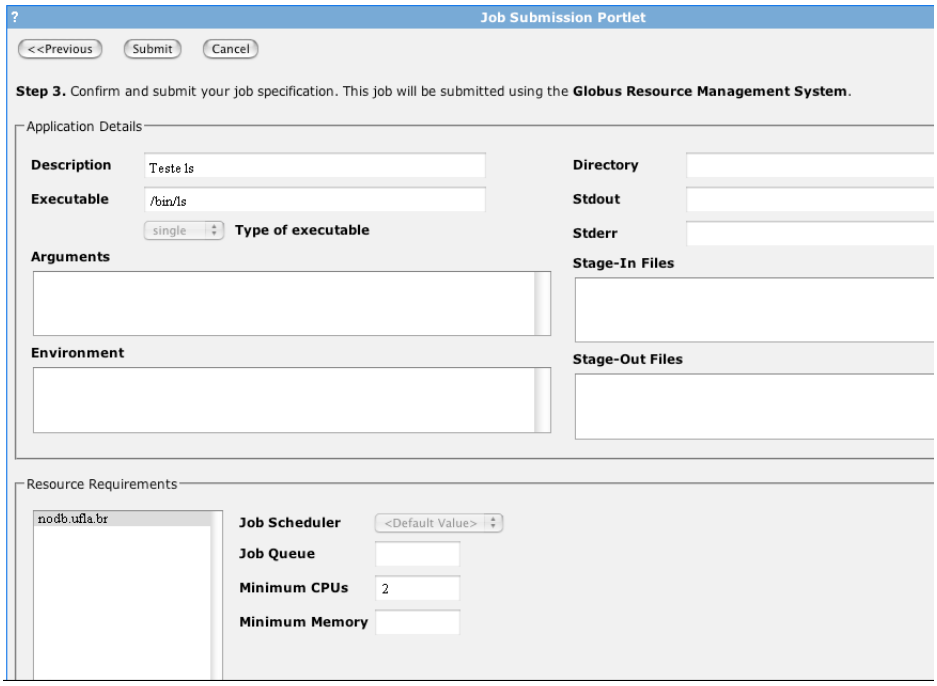

Figura 4.5: Confirmação do submissão do Teste ls.

Após a execução, o resultado pode ser visualizado no campo *Job Output*, todos os arquivos encontrados no diretório *home* do usuário (Figura 4.6). Pode-se verificar que são os mesmos arquivos listados ao ser executado o comando via terminal.

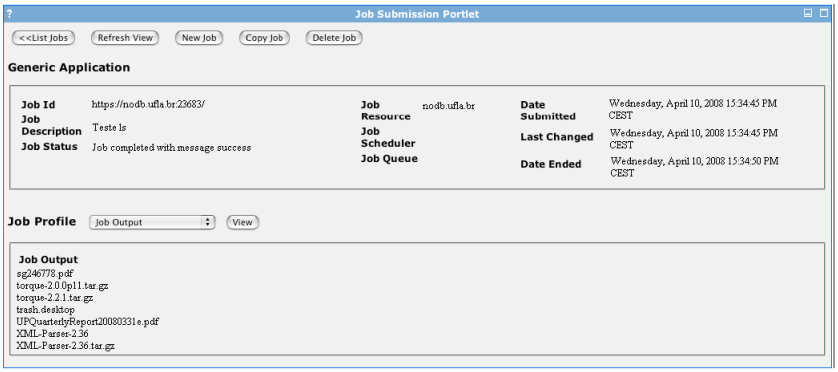

Figura 4.6: Resultado da submissão do Teste ls.

4.3.2 Submissão data

Esta tarefa tem como objetivo fazer um *sleep* de aproximadamente 60 segundos e após este tempo retornar a data e a hora do sistema. Primeiramente a submissão foi realizada via o WS GRAM:

```
[nemesio@nodb ~]$ globusrun-ws -submit -F
 https://nodb.ufla.br:8443/wsrf/ManagedJobFactory -Ft PBS -c echo "sleep 60;date"
Submitting job...Done.
Job ID: uuid:547f0fd6-a4b6-11da-ac33-0011d8b1eb22
Termination time: 02/09/2008 17:09 GMT
Current job state: Pending
Current job state: Active
.......Saída..............
Thu Feb 9 17:09:53 CST 2008
........................
Current job state: CleanUp
Current job state: Done
Destroying job...Done.
```

Ao término deste submissão, irá ser feita a submissão da tarefa via portal *Grid*. Como no item anterior foi apresentado a submissão mais detalhadamente. Neste item terá a submissão propriamente dita, pois a parte de entrar com a nova credencial já foi explica no item anterior. A Figura 4.7 mostra como devemos preencher os dados da submissão principalmente com sua descrição e o comando para ser executado pelo portal:

No segundo passo deve-se escolher em qual máquina o *job* será executado. Como exemplo foi utilizado uma máquina apenas, e podemos escolher qual gerenciador de *jobs* irá ser utilizado para submeter a tarefa (Figura 4.8).

Novamente, como é uma tarefa simples, não foi preciso fazer nenhum upload de script, bastou colocar o próprio comando no campo certo (Figura 4.9):

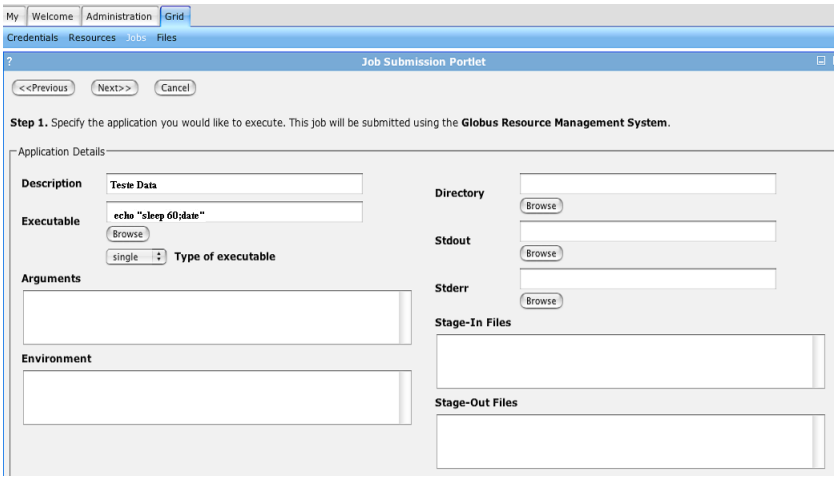

Figura 4.7: Início da submissão do Teste data.

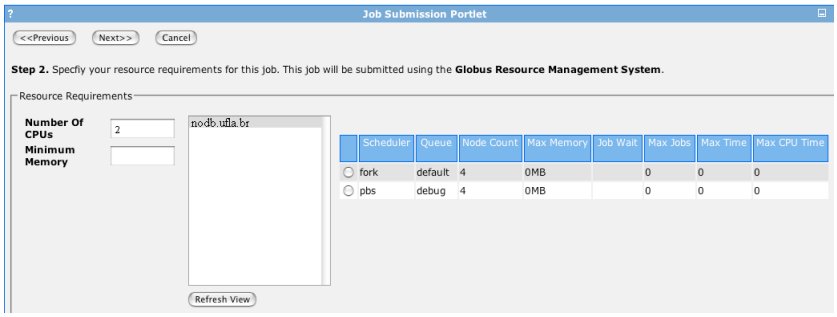

Figura 4.8: Escolha da máquina a qual irá executar a tarefa data.

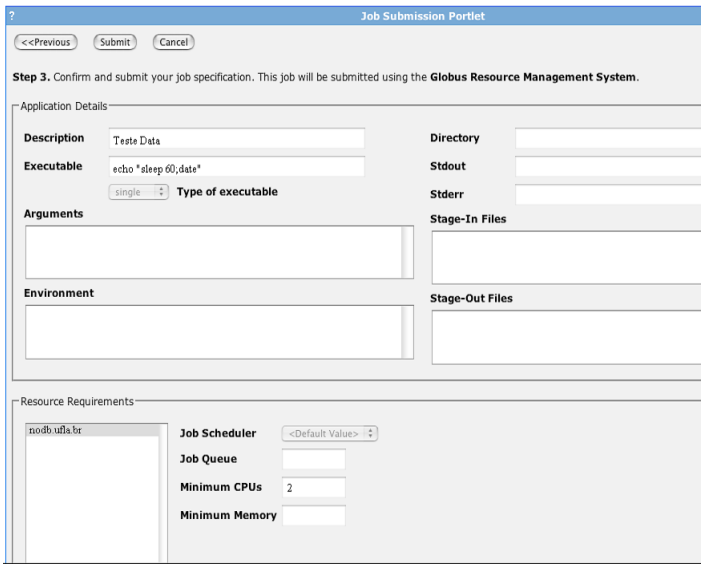

Figura 4.9: Confirmação do submissão do Teste data.

Como resultado, pode-se ver a data do sistema e as horas, notando que a tarefa demorou 60 segundos para executar devido ao comando *sleep* (Figura 4.10).

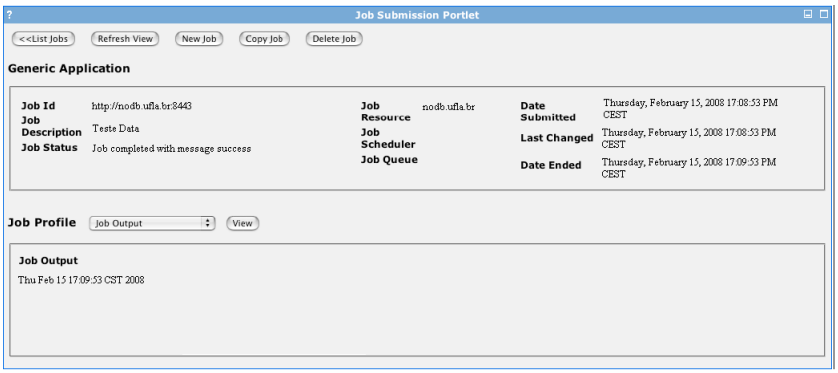

Figura 4.10: Resultado da submissão do Teste data.

4.3.3 Submissão *echo_job*

Para a submissão deste *job* foi criado um pequeno *script*. Este teste foi realizado para mostrar os diversos tipos de submissão que podem existir, como existem milhares. Tentou-se mostrar os mais comuns.

No primeiro passo deve-se criar um *script*, em um editor de texto, colocar o conteúdo abaixo e salvá-lo como .xml:

```
<?xml version="1.0" encoding="UTF-8"?>
<job>
<executable>/bin/echo</executable>
<argument> Este job tem a finalidade de testar o ambiente grid.</argument>
<stdout>${GLOBUS_LOCATION }/stdout</stdout>
<stderr>${GLOBUS_LOCATION}/stderr</stderr>
</job>
```

O *script* criado tem nada mais que a função de fazer um echo com a frase "Este job tem a finalidade de testar o ambiente grid". No final do *script* foi indicado o nome dos arquivos para saída padrão e saída de erros, tanto o WS GRAM quanto o portal já exibem as saídas e os erros. Porém, para resultados muito extensos estes arquivos de erros e saídas são necessários.

A execução com o WS GRAM foi realizado com o comando abaixo:

```
[nemesio@nodb ~]$ globusrun-ws -submit -F
https://nodb.ufla.br:8443/wsrf/ManagedJobFactoryService -Ft PBS -c
/home/nemesio/echo_job.xml
Submitting job...Done.
```

```
Job ID: uuid:547f0fd6-a4b6-11da-ac33-0011d8b1eb22
Termination time: 04/10/2006 14:34 GMT
Current job state: Pending
Current job state: Active
........Saída..............
Este job tem a finalidade de testar o ambiente grid.
.........................
Current job state: CleanUp
Current job state: Done
Destroying job...Done.
```

Após a execução via WS GRAM e obter o resultado, o mesmo teste foi efetuado utilizando o portal *Grid*. Obeserve na Figura 4.11 que deve-se colocar a descrição da tarefa, porém neste exemplo como estamos submetendo um script não coloca-se nada no campo *Executable* mas sim o diretório onde está contido o *script*.

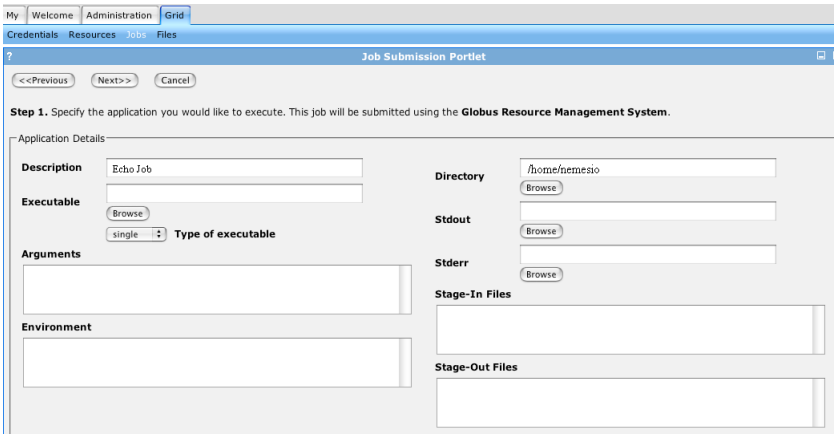

Figura 4.11: Início da submissão do *echo_job*.

No segundo passo foi escolhida em qual máquina o job será executado e qual gerenciador de *jobs* será escolhido para submeter o *script*. No exemplo foi utilizada uma máquina apenas (Figura 4.12).

Neste exemplo, como foi utilizado um *script*, deve-se verificar o diretório em que o script se encontra (Figura 4.13).

Ao executar pode-se observar que a submissão obteve o mesmo resultado, fazendo um *echo* na frase "Este job tem a finalidade de testar o ambiente grid", resultado igual quando a submissão foi feita através do WS GRAM (Figura 4.14).

Note que como foi definida a criação de dois arquivos um de erro e um de saída e como não houve nenhum erro na execução, o arquivo de erro está vazio. Somente o arquivo de saída tem algum conteúdo. Se tivesse ocorrido algum erro durante a execução, seria o arquivo de saída que estaria vazio.

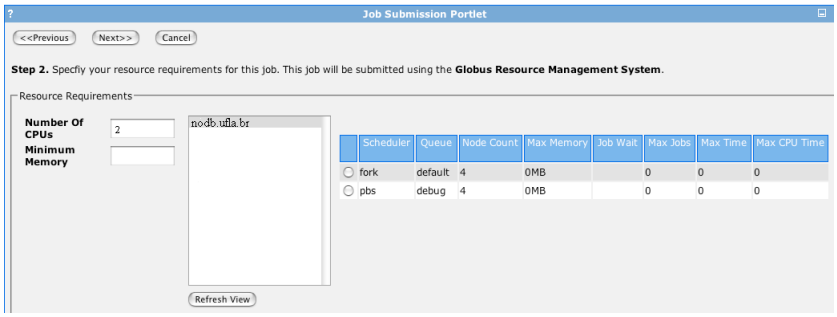

Figura 4.12: Escolha da máquina a qual ira ser executada a tarefa *echo_job*.

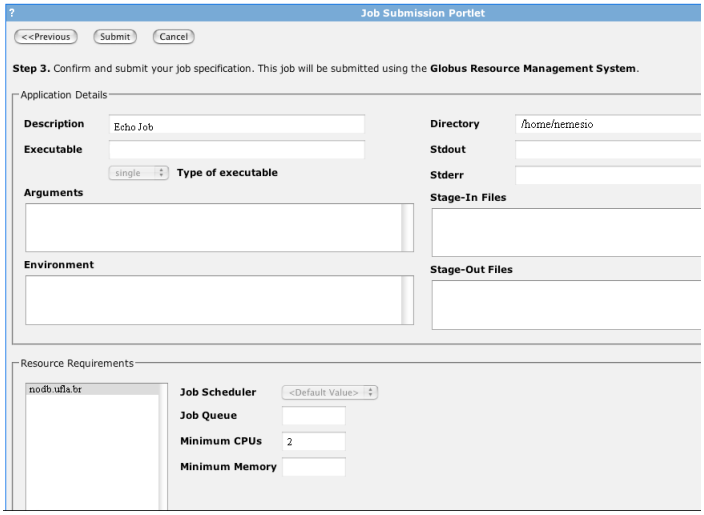

Figura 4.13: Confirmação do submissão do *echo_job*.

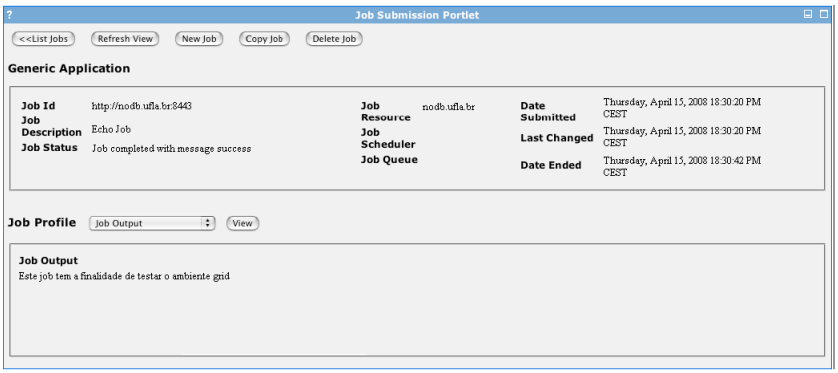

Figura 4.14: Resultado da submissão do *echo_job*.

4.3.4 Submissão *multiple_job*

Esta submissão é muito parecida com a anterior a *echo_job*, porém apresenta um echo de uma frase com múltiplos *jobs*, e duas máquinas irão responder um *echo*, testando assim o multiparalelismo do

ambiente.

Como no exemplo anterior, foi criado um *script* salvo como .xml. Este *script* é um *script* básico de paralelismo de *jobs* onde é passado os endereços das máquinas que irão executar as tarefas, os arquivos das saídas padrões e de erros e executável da tarefa:

```
<?xml version="1.0" encoding="UTF-8"?>
<multiJob xmlns:gram="http://www.globus.org/namespaces/2004/10/gram/job"
xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/03/addressing">
<job>
<factoryEndpoint>
<wsa:Address>https://noda.ufla.br:8443/wsrf/services/ManagedJobFactory
Service</wsa:Address>
<wsa:ReferenceProperties>
<gram:ResourceID>Fork</gram:ResourceID>
</wsa:ReferenceProperties>
</factoryEndpoint>
<executable>/bin/echo</executable>
<argument>This file is the first file written by WS GRAM job with
multiple job definition file.</argument>
<stdout>${GLOBUS_USER_HOME}/stdout_multi_1</stdout>
<stderr>${GLOBUS_USER_HOME}/stderr_multi_1</stderr>
<count>2</count>
</job>
<job>
<factoryEndpoint>
<wsa:Address>https://nodb.ufla.br:8443/wsrf/services/ManagedJobFactory
Service</wsa:Address>
<wsa:ReferenceProperties>
<gram:ResourceID>Fork</gram:ResourceID>
</wsa:ReferenceProperties>
</factoryEndpoint>
<executable>/bin/echo</executable>
<argument>This file is the second file written by WS GRAM job with
multiple job definition file.</argument>
<stdout>${GLOBUS_USER_HOME}/stdout_multi_2</stdout>
<stderr>${GLOBUS_USER_HOME}/stderr_multi_2</stderr>
<count>1</count>
</job>
</multiJob>
```

Neste exemplo os *sub-jobs* serão enviados para duas máquinas específicas. No teste realizado foram utilizadas duas máquinas. Em um ambiente maior pode-se colocar de uma forma que as maquinas

disponíveis recebam os *jobs*, de acordo com sua prioridade.

Primeiramente, o *script* foi executado via WS GRAM.

```
[nemesio@nodb ~]$ globusrun-ws -submit -F
https://nodb.ufla.br:8443/wsrf/ManagedJobFactoryService
-Ft PBS -c /home/nemesio/multi_job.xml
Submitting job...Done.
Job ID: uuid:547f0fd6-a4b6-11da-ac33-0011d8b1eb22
Termination time: 09/04/2008 16:19 GMT
Current job state: Pending
Current job state: Active
.......Saída.............
Job teste, com a finalidade de testar a função de múltiplos jobs do ambiente
Job teste, com a finalidade de testar a função de múltiplos jobs do ambiente
.........................
Current job state: CleanUp
Current job state: Done
Destroying job...Done.
```

Após a execução via WS GRAM, o mesma tarefa foi executada utilizando o portal *Grid*, veja na Figura 4.15 que temos que colocar a descrição da tarefa, porém neste exemplo como estamos submetendo um *script* não devemos colocar nada no campo *Executable* mas sim colocar o diretório que está contido o *script*.

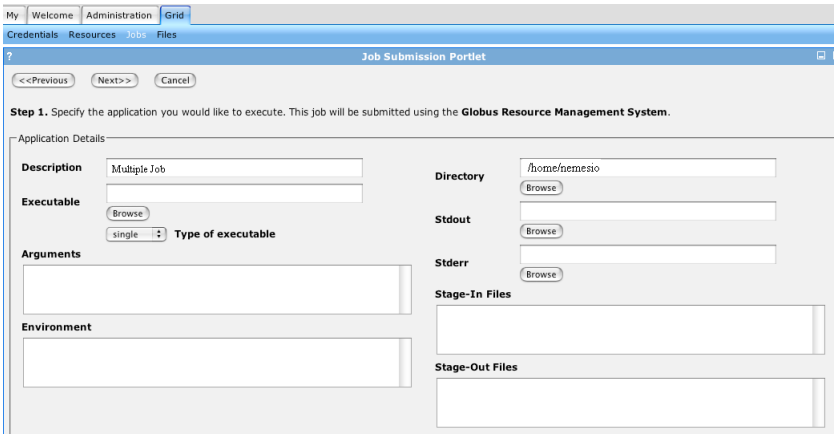

Figura 4.15: Início da submissão do *multiple_job*.

No segundo passo foi escolhido qual máquina o job será executado e qual gerenciador de jobs será escolhido para submeter o *script*. Como este exemplo faz submissão de *multi job´s*, teremos no portal que ter no mínimo duas máquinas ativas para ser efetuada a execução da tarefa (Figura 4.16).

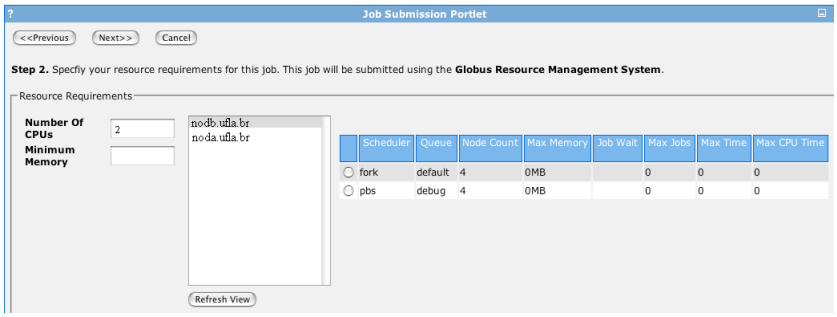

Figura 4.16: Escolha da máquina a qual irá ser executada a tarefa *Multiple Job*.

Neste exemplo, como estamos utilizando um *script*, deve-se verificar o diretório em que o *script* se encontra (Figura 4.17).

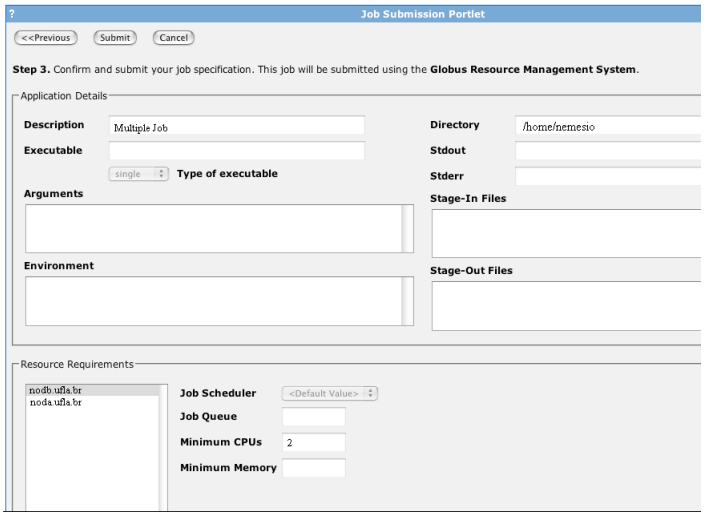

Figura 4.17: Confirmação da submissão do *multiple_job*.

Como resultado tem-se o *echo* de duas frases, onde que cada uma foi gerada pela máquina específica, podendo assim concluir a eficiência do multiparalelismo do ambiente (Figura 4.18).

4.4 Considerações Finais

Ao final desta seção pode-se verificar a eficiência do ambiente *Grid* juntamente com seus serviços, e os benefícios que o portal *Grid* pode trazer ao ambiente. O portal além de facilitar a vida dos usuários e dos administradores, ele encapsula muitos aspectos complexos do ambiente, trazendo assim uma eficiência maior ao ambiente *Grid*.

Este capítulo apresentou alguns testes do ambiente *Grid* que foi instalado e configurado para apenas 2 máquinas. Para aproveitar melhor as funcionalidades de um ambiente *Grid* é necessário que haja

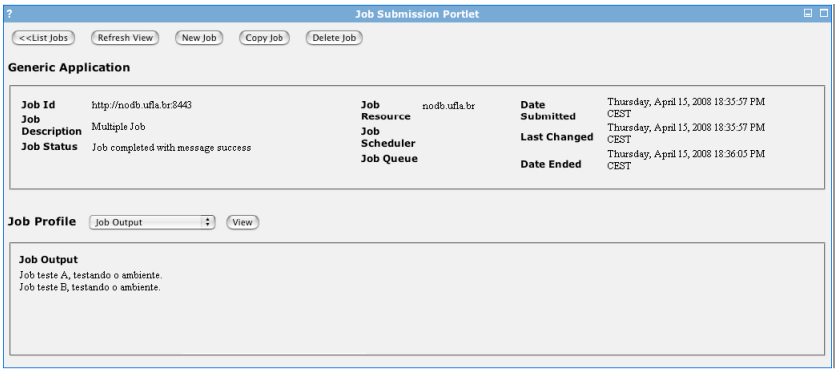

Figura 4.18: Resultado da submissão do *multi_job*.

um número maior de recursos computacionais. Os testes de submissão de *jobs* foram realizados para exemplos bem simples tentando contemplar submissões com e sem *scripts*.

A submissão de tarefas mais complexas necessita de um *script* e consequentemente de um conhecimento mais aprofundado da linguagem de descrição de *jobs* utilizado pelo *Globus Toolkit*, que não foram abordados neste trabalho por não ser o seu foco.

Capítulo 5

Conclusões e Trabalhos Futuros

A tecnologia *Grid* é uma tecnologia nova que permite o compartilhamento de recursos computacionais de forma integrada e segura.

Um ambiente em *Grid* contém grandes benefícios em comparação com um ambiente em *cluster*. Mas apresenta também uma complexidade maior, podendo tornar o acesso aos serviços e recursos do *Grid* muito difícil tanto para administradores quanto para usuários genéricos. Diante deste cenário é preciso esconder alguns detalhes do ambiente em *Grid*, para tentar deixá-lo um pouco mais acessível aos usuários e mais fácil de gerenciar.

Neste trabalho, foram apresentadas as principais características e vantagens que a utilização da tecnologia *Grid* pode trazer tanto aos usuários comuns como às grandes organizações. Pode-se concluir que a tecnologia *Grid* finalmente saiu do papel e tem tomado o seu lugar nas grandes instituições de ensino, e também em grandes empresas e organizações, visando assim o mundo dos negócios. Podemos destacar algumas empresas que hoje em dia já utilizam essa nova tecnologia como a Oracle, a IBM, a Microsoft entre outras.

Porém, toda tecnologia nova possui algumas dificuldades de ser implantada, pois é difícil em uma grande organização quebrar paradigmas de um dia para o outro, mudar de tecnologias antigas estáveis para tecnologias novas e instáveis, mas o objetivo final é sempre obter benefícios.

Visando apresentar a implantação desta tecnologia, este trabalho teve como principal objetivo, a implantação completa do *middleware Globus Toolkit* 4 (GT4) em duas máquinas do Laboratório de Pesquisa Científica do DCC/UFLA com foco em segurança e usabilidade. Ao final do trabalho foi gerado um manual de fácil uso e entendimento sobre a instalação e implantação do GT4. O único pré-requisito existente é que qualquer usuário que venha a colocar este manual em prática tenha o mínimo de conhecimento sobre o sistema operacional Linux.

O *Globus Toolkit* oferece vários serviços. Neste trabalho, foram enfatizados a usabilidade e a segurança. Por isso foram colocadas em prática os principais serviços do *Globus Toolkit* como o GridFTP,

SimpleCA, Web service Container, RFT, conexão com o Postgresql e WS GRAM.

Para melhorar a usabilidade do GT4 foi realizada a integração do *Globus Toolkit*, com o portal *Grid* utilizando o *Gridsphere*. Além disso foi feita a integração com o gerenciador de *jobs Torque (OpenPBS)*.

Os testes realizados mostraram que *jobs* com diferentes características podem ser executados neste ambiente.

O GT4 tem diversos serviços que podem ser melhor explorados e possui potencial para realizar integrações com outras entidades de ensino e pesquisa.

Como trabalhos futuros pode-se realizar a criação de um ambiente de *Grid* interligando os laboratórios de pesquisa do DCC/UFLA com outros laboratórios de pesquisa da UFLA e até mesmo de outras instituições de ensino. Esta interligação em um ambiente de *Grid* poderá gerar o compartilhamento de recursos computacionais e formação de parcerias para diversas pesquisas.

Outro trabalho futuro que poderá ser realizado é a integração do GT4 com outros gerenciadores de *jobs* como o SGE, CONDOR e o LSF. Para melhorar a interface com o usuário, o portal *Grid* poderá ser aprimorado oferecendo mais informações sobre os nós e os *jobs* que estão sendo executados e até mesmo a criação de *wiki´s* para possibilitar a troca de experiências dos usuários.

Referências Bibliográficas

[1] Condor project. Disponível em: http://www.cs.wisc.edu/condor/ Acesso em: Março 2008.

[2] Debian. Disponível em: http://www.debian.org. Acesso em: Março 2008.

[3] The easy grid project. Disponível em: http://easygrid.ic.uff.br/ Acesso em: Maio 2008.

[4] Fedora core linux. Disponível em: http://fedoraproject.org/. Acesso em: Março 2008.

[5] glite middleware. Disponível em: http://glite.web.cern.ch/glite/ Acesso em: Maio 2008.

[6] Gridsphere portlet reference guide. Disponível em http://www.gridsphere.org/gridsphere/docs/ Acesso em: Fevereiro 2008.

[7] Https. Disponível em: http://pt.wikipedia.org/wiki/HTTPS. Acesso em: Março 2008.

[8] Ibm. Disponível em: http://www.ibm.com/br/. Acesso em: Março 2008.

[9] Oracle. Disponível em: http://www.oracle.com/database/index.html. Acesso em: Março 2008.

[10] Pbs. Disponível em: http://www.pbs.org/. Acesso em: Março 2008.

[11] Perl. Disponível em: http://www.perl.org/. Acesso em: Março 2008.

[12] Red hat linux. Disponível em:http://www.redhat.com/. Acesso em: Março 2008.

[13] Sistema nacional de processamento de alto desempenho / ministério da ciência e tecnologia rede pública de centros de pad. Disponível em: http://www.lncc.br/sinapad/ Acesso em: Maio 2008.

[14] Sun. Disponível em: http://br.sun.com/. Acesso em: Março 2008.

[15] Sun grid engine. Disponível em:http://www.sun.com/software/gridware Acesso em: Março 2008.

[16] Suse linux. Disponível em: http://www.novell.com/linux/. Acesso em: Março 2008.

[17] T. G Alliance. Globus toolkit 4.0 execution management. Disponível em: http://www.globus.org/toolkit/docs/4.0/execution/. Acessado em: Novembro 2007.

[18] T. G Alliance. Globus toolkit 4.0 ws gram. Disponível em: http://www.globus.org/toolkit/docs/4.0/execution/wsgram/. Acessado em: Novembro 2007.

[19] T. G Alliance. Globus toolkit 4.0 ws gram documento pdf. Disponível em: http://www.globus.org/toolkit/docs/4.0/execution/wsgram/index.pdf. Acessado em: Novembro 2007.

[20] T. G Alliance. Globus toolkitt 4.0. Disponível em: http://www.globus.org/toolkit/docs/4.0/. Acessado em: Novembro 2007.

[21] V BERSTIS. Fundamentals of grid computing. Disponível em: http://www.redbooks.ibm.com/redpapers/pdfs/redp3613.pdf. Acesso em: Fevereiro 2008.

[22] F. BOMBONATO. Computação em grid. uma introdução. Disponível em: http://www.geleira.org/pdf/grid_computing.pdf. Acesso em: Fevereiro 2008.

[23] Morgan Kaufmann Borja Sotomayor, Lisa Childers. *Globus Toolkit 4: Programming Java Services*. First Edition, December 16, 2005.

[24] E Cerami. Web services essentials. O Reilly, Vol. IV 2005.

[25] M. DANTAS, J. ALLEMAND, and L PASSOS. An evaluation of globus and legion software environments. Disponível em: http://hpcs2003.ccs.usherbrooke.ca/papers/Dantas_01.pdf. Acesso em: Fevereiro 2008.

[26] I. FOSTER and C. KESSELMAN. Globus: A metacomputing infrastructure toolkit. International Journal of Supercomputer Aplications 11, 2, 115-128. Disponível em: ftp://ftp.globus.org/pub/globus/papers/globus.pdf. Acesso em: Fevereiro 2008.

[27] I. FOSTER, C. KESSELMAN, and S. TUECKE. The anatomy of the grid: Enabling scalable virtual organizations. Disponível em: http://www.globus.org/research/papers/anatomy.pdf. Acesso em: Fevereiro 2008.

[28] Globus. Globus toolkit. Disponível em: http://www.globus.org/ Acesso em: Março 2008.

[29] GPDK. Grid portal development kit. Disponível em: http://doesciencegrid.org/projects/GPDK/. Acesso em: Novembro 2007.

[30] Legion. Legion faq. Disponível em: http://legion.virginia.edu/FAQ.html. Acesso em: Fevereiro 2008.

[31] BAKER M., BUYYA R., and LAFORENZA D. Grids and grid technologies for widearea distributed computing. *Software Practice and Experience.*, 2002.

[32] Michael Miley. The grid: Bringing computing power to the masses. Stokie: Oracle Magazine, número 5, Vol. XVII, pag. 39-44, 2003.

[33] R. D. S. Moreira. Certificados digitais. Disponível em: http://www.gta.ufrj.br/grad/00 1/rodrigo/fr9right.htm. Acesso em: Novembro 2007.

[34] M. NASH. Oracle 10g: Infrastructure for grid computing. Disponível em:Disponível em: http://otn.oracle.com/tech/grid/collateral/GridTechWhitePaper_final.pdf. Acesso em: Fevereiro 2008.

[35] A. NATRAJAN, A. NGUYEN-TUONG, M. HUMPHREY, and A. GRIMSHAW. The legion portal. Disponível em: http://legion.virginia.edu/papers/GCE01.pdf. Acesso em: Fevereiro 2008.

[36] J. NOVOTNY, M. RUSSEL, and O. WEHRENS. Gridsphere: An advanced portal framework. Disponível em: http://www.gridsphere.org/gridsphere/wp-4/Documents/France/gridsphere.pdf. Acesso em: Novembro 2007.

[37] OASIS. Oasis web service resource framework. Disponível em: http://www.oasis-open.org/home/index.php. Acesso em: Março 2008., 2006.

[38] Marluce Rodrigues Pereira Patrícia K. Vargas. Computação em grade. Escola Regiona de Informática de Minas Gerais, 2007,UFLA-MG.

[39] Marcos Pitanga. Computação em cluster-o estado da arte em computação. Rio de Janeiro: Editora Brasport, 2003.

[40] SETI@home. Search for extraterrestrial intelligence. Disponível em: http://setiathome.berkeley.edu/. Acesso em: Fevereiro 2008.

[41] B. Sotomayor. The globus toolkit programmers tutorial. Disponível em: http://gdp.globus.org/gt4-tutorial/. Acesso em: Fevereiro 2008.

[42] OASIS Standard. Web services resource 1.2 (ws-resource). Organization for the Advancement of Structured Information Standards.

[43] M. THOMAS, M. DAHAN, K. MUELLER, C. MOCK, S.and MILLS, and R. REGNO. Application portals: Practice and experience. Disponível em: http://tacc.utexas.edu/ mthomas/pubs/GridPort_Apps_CPE.pdf. Acesso em: Fevereiro de 2008.

Apêndice A

Apêndice A - Pré-Requisitos para Instalação do *Globus Toolkit* 4

A.1 Criando Certificado de Autorização

Passo a passo da instalação do certificado de autorização para a criação da unidade certificadora.

```
[globus@nodeB gt4.0.1-all-source-installer]$
$GLOBUS_LOCATION/setup/globus/setup-simple-ca

WARNING: GPT_LOCATION not set, assuming:
GPT_LOCATION=/opt/globus-4.0.1

C e r t i f i c a t e A u t h o r i t y S e t u p

This script will setup a Certificate Authority for signing Globus users certificates.
It will also generate a simple CA package that can be distributed to the users of the CA.

The CA information about the certificates it distributes will be kept in:

/home/globus/.globus/simpleCA/

The unique subject name for this CA is:

cn=Globus Simple CA, ou=simpleCA-nodeb.ps.univa.com, ou=GlobusTest, o=Grid

Do you want to keep this as the CA subject (y/n) [y]:

n
```

```
Enter a unique subject name for this CA:

cn=<MyOrganization>,ou=ConsortiumTutorial,ou=GlobusTest,o=Grid

Enter the email of the CA
(this is the email where certificate requests will be sent to be signed by the CA):

<MyEmailAddress>

The CA certificate has an expiration date.
Keep in mind that once the CA certificate has expired.
All the certificates signed by that CA, become invalid.
This can be done by re-running this setup script.
Enter the number of DAYS the CA certificate should last before it expires.
[default: 5 years (1825 days)]:

Enter PEM pass phrase:<MyPEMpassPhrase>
Verifying - Enter PEM pass phrase:<MyPEMpassPhrase>
creating CA config package...done.

A self-signed certificate has been generated
for the Certificate Authority with the subject:

/O=Grid/OU=GlobusTest/OU=ConsortiumTutorial/CN=<MyOrganization>

If this is invalid, rerun this script

/opt/globus-4.0.1/setup/globus/setup-simple-ca

and enter the appropriate fields.

---------------------------------------------------------------------

The private key of the CA is stored in
/home/globus/.globus/simpleCA//private/cakey.pem

The public CA certificate is stored in
/home/globus/.globus/simpleCA//cacert.pem

The distribution package built for this CA is stored in

/home/globus/.globus/simpleCA//globus_simple_ca_768c46a5_setup-0.19.tar.gz
```

```
This file must be distributed to any host wishing to request
certificates from this CA.

CA setup complete.

The following commands will now be run to setup the security
configuration files for this CA:

$GLOBUS_LOCATION/sbin/gpt-build

/home/globus/.globus/simpleCA//globus_simple_ca_768c46a5_setup-0.19.tar.gz

$GLOBUS_LOCATION/sbin/gpt-postinstall
------------------------------------------------------------------
setup-ssl-utils: Configuring ssl-utils package
Running setup-ssl-utils-sh-scripts...

Note: To complete setup of the GSI software you need to run the following script as root
to configure your security configuration directory:

globus-4.0.1/setup/globus_simple_ca_f1f2d5e6_setup/setup-gsi

For further information on using the setup-gsi script, use the -help option.

The -default option sets this security configuration to be the default,
and -nonroot can be used on systems where root access is not available.
setup-ssl-utils: Complete
```

A.2 Implantação do *PostgreSQL Relational Database*

Para o funcionamento do Globus é necessária a instalação do Reliable File Transfer (RFT), dependendo da versão do Fedora Core o PostgreSQL já pode estar instalado e configurado. Porém para verificar a sua instalação basta utilizar os comandos abaixo:

```
[root@nodeB ~]# rpm -qa|grep postgres
```

O Resultado retornado deve ser semelhante a este:

```
postgresql-8.0.3-1
postgresql-server-8.0.3-1
postgresql-libs-8.0.3-1
```

É necessário que os os três pacotes estejam instalados. Caso não estejam pode-se usar 'yum' para instalar os pacotes. Para garantir que o ´postgres´ está disponível basta executar o comando abaixo:

```
[root@nodeB ~]# grep postgres /etc/passwd
postgres:x:26:26:PostgreSQL Server:/var/lib/pgsql:/bin/bash
```

O Postgres não deve iniciar automaticamente. Para verificar se está iniciando automaticamente basta utilizar o comando ´chkconfig '. Se estiver com tudo off é porque não está iniciando automaticamente.

```
[root@nodeB ~]# /sbin/chkconfig --list | grep postgres
postgresql 0:off 1:off 2:off 3:off 4:off 5:off 6:off
```

Em seguida, é preciso inicializar o PostgreSQL database. Foi utilizado o local não-padrão para arquivos de dados PostgreSQL, a fim de prevenir eventuais problemas com bases anteriormente utilizadas.

Como root, é necessário criar um diretório que é propriedade do usuário postgres no grupo postgres, e em seguida realizar login com o usuário "postgres".

```
[root@nodeB ~]# mkdir -p /opt/pgsql/data
[root@nodeB ~]# chown -R postgres /opt/pgsql
[root@nodeB ~]# chgrp -R postgres /opt/pgsql
[root@nodeB ~]# su - postgres
```

Em seguida, é preciso inicializar o banco de dados, com a opção-D para apontar para o local para onde os novos arquivos de dados serão armazenados. O comando e suas mensagens de saída são mostradas abaixo:

```
-bash-3.00$ /usr/bin/initdb -D /opt/pgsql/data

The files belonging to this database system will be owned by user "postgres".
This user must also own the server process.
The database cluster will be initialized with locale en_US.UTF-8.
The default database encoding has accordingly been set to UNICODE.
fixing permissions on existing directory /opt/pgsql/data ... ok
creating directory /opt/pgsql/data/global ... ok
creating directory /opt/pgsql/data/pg_xlog ... ok
creating directory /opt/pgsql/data/pg_xlog/archive_status ... ok
creating directory /opt/pgsql/data/pg_clog ... ok
creating directory /opt/pgsql/data/pg_subtrans ... ok
creating directory /opt/pgsql/data/base ... ok
creating directory /opt/pgsql/data/base/1 ... ok
creating directory /opt/pgsql/data/pg_tblspc ... ok
```

```
selecting default max_connections ... 100
selecting default shared_buffers ... 1000
creating configuration files ... ok
creating template1 database in /opt/pgsql/data/base/1 ... ok
initializing pg_shadow ... ok
enabling unlimited row size for system tables ... ok
initializing pg_depend ... ok
creating system views ... ok
loading pg_description ... ok
creating conversions ... ok
setting privileges on built-in objects ... ok
creating information schema ... ok
vacuuming database template1 ... ok
copying template1 to template0 ... ok
WARNING: enabling "trust" authentication for local connections
You can change this by editing pg_hba.conf or using the
-A option the next time you run initdb.
Success. You can now start the database server using:

/usr/bin/postmaster -D /opt/pgsql/data
or
/usr/bin/pg_ctl -D /opt/pgsql/data -l logfile start

The database is now initialized.
```

Em seguida, orientar stdout(log de saída) e stderr(log de erro) para um arquivo de log e iniciar o banco de dados em segundo plano:

```
-bash-3.00$
/usr/bin/postmaster -i -D /opt/pgsql/data > /opt/pgsql/logfile 2>&1 &
```

A.3 Instalando Java e ant

Os pacotes java instalados para Fedora Core 6 devem ser removidos, pois podem causar incompatibilidades com o *Globus Toolkit* 4. Estes pacotes foram removidos e foi instalada a versão mais recente do JDK.

```
[root@nodeB ~]# which java //Comando para verificar se o java está instalado
/usr/bin/java
```

Esse arquivo é normalmente uma ligação simbólica:

```
[root@nodeB ~]# ls -alh /etc/alternatives/java
lrwxrwxrwx 1 root root 35 Oct 7 16:06 /etc/alternatives/java
-> /usr/lib/jvm/jre-1.4.2-gcj/bin/java

[root@nodeB ~]# rpm -qf /usr/lib/jvm/jre-1.4.2-gcj/bin/java
java-1.4.2-gcj-compat-1.4.2.0-40jpp_31rh.FC4.2  // Saída do comando
```

Com o nome do pacote binário conhecido, basta utilizar yum para removê-lo:

```
[root@nodeB ~]# yum remove java-1.4.2-gcj-compat-1.4.2.0-40jpp_31rh
```

O link indicado para se efetuar o download do JDK é http://java.sun.com/j2se/1.5.0/download.jsp. Clique para fazer o download aceitando os termos de compromisso e licença.

```
[root@nodeB ~]# ls -l jdk-1_5_0_06-linux-i586.bin
-rw-r--r-- 1 root root 48974825 Feb 20 11:13 jdk-1_5_0_06-linux-i586.bin
```

O md5 checksum para o arquivo é:

```
[root@nodeB ~]# md5sum jdk-1_5_0_06-linux-i586.bin
3cdad4a383b93680f02f6f06198c2227 jdk-1_5_0_06-linux-i586.bin
```

Mova o arquivo para o diretório /opt e de permissão de execução.(obs: o arquivo poderia ser mandado para qualquer diretório, o /opt foi uma escolha que fizemos)

```
[root@nodeB ~]# mv jdk-1_5_0_06-linux-i586.bin /opt/
[root@nodeB ~]# cd /opt/
[root@nodeB opt]# chmod 755 jdk-1_5_0_06-linux-i586.bin
```

Vamos executar o arquivo e descompactar sua instalação:

```
[root@nodeB opt]# ./jdk-1_5_0_06-linux-i586.bin
```

Após o comando você precisará ler toda a pagina de licença e por final responder "SIM"para aceitar os termos de licença, após isso o conteúdo do arquivo será descompactado:

```
[root@nodeB opt]# ls -l /opt/jdk1.5.0_06/
total 17284
drwxr-xr-x 2 root root 4096 Nov 10 16:19 bin
-r--r--r--1 root root 2487 Nov 10 15:38 COPYRIGHT
drwxr-xr-x 8 root root 4096 Nov 10 16:19 demo
drwxr-xr-x 3 root root 4096 Nov 10 16:19 include
drwxr-xr-x 6 root root 4096 Nov 10 16:19 jre
drwxr-xr-x 2 root root 4096 Feb 20 11:16 lib
```

```
-r--r--r--1 root root 15584 Nov 10 15:38 LICENSE
drwxr-xr-x 4 root root 4096 Nov 10 16:19 man
-r--r--r--1 root root 20415 Nov 10 15:38 README.html
drwxr-xr-x 4 root root 4096 Nov 10 16:19 sample
-rw-r--r--1 root root 17527615 Nov 10 15:38 src.zip
-r--r--r--1 root root 68419 Nov 10 15:38 THIRDPARTYLICENSEREADME.txt
```

Vamos testar a instalação do java executando:

```
[root@nodeB opt]# /opt/jdk1.5.0_06/bin/java -version
java version "1.5.0_06"
Java(TM) 2 Runtime Environment, Standard Edition (build 1.5.0_06-b05)
Java HotSpot(TM) Client VM (build 1.5.0_06-b05, mixed mode, sharing)
```

A mesma coisa para o compilador java:

```
[root@nodeB opt]# /opt/jdk1.5.0_06/bin/javac -version
javac 1.5.0_06
javac: no source files
Usage: javac <options> <source files>
```

Por fim temos que exportar as variáveis de ambiente do java em referencia ao seu local de instalação e ao seu diretório /bin, segue a baixo:

```
[root@nodeB ~]# export JAVA_HOME=/opt/jdk1.5.0_06
[root@nodeB ~]# export PATH=$JAVA_HOME/bin:$PATH
[root@nodeB ~]# which java
/opt/jdk1.5.0_06/bin/java
[root@nodeB ~]# which javac
/opt/jdk1.5.0_06/bin/javac
```

Agora iniciaremos a instalação do ANT, podemos baixar o ANT através do link:
http://apache.mirrormax.net/ant/binaries/apache-ant-1.6.5-bin.tar.bz2

```
[root@nodeB opt]# ls -l apache-ant-1.6.5-bin.tar.bz2
-rw-r--r-- 1 root root 6743024 Jun 2 2005 apache-ant-1.6.5-bin.tar.bz2
```

O md5 checksum do arquivo é :

```
[root@nodeB opt]# md5sum apache-ant-1.6.5-bin.tar.bz2
26031ee1a2fd248ad0cc2e7f17c44c39 apache-ant-1.6.5-bin.tar.bz2
```

Iremos mover o arquivo para o diretório /opt e descompactá-lo:

```
[root@nodeB ~]# mv apache-ant-1.6.5-bin.tar.bz2 /opt/
[root@nodeB ~]# cd /opt/
[root@nodeB opt]# tar -jxf apache-ant-1.6.5-bin.tar.bz2
```

Apos descompactá-lo ele terá os seguintes arquivos instalados:

```
[root@nodeB opt]# ls -l apache-ant-1.6.5
total 208
drwxr-xr-x 2 root root 4096 Feb 20 11:32 bin
drwxr-xr-x 6 root root 4096 Feb 20 11:32 docs
drwxr-xr-x 3 root root 4096 Feb 20 11:32 etc
-rw-r--r--1 root root 126 Jun 2 2005 INSTALL
-rw-r--r--1 root root 17191 Jun 2 2005 KEYS
drwxr-xr-x 2 root root 4096 Feb 20 11:32 lib
-rw-r--r--1 root root 11766 Jun 2 2005 LICENSE
-rw-r--r--1 root root 3356 Jun 2 2005 LICENSE.dom
-rw-r--r--1 root root 677 Jun 2 2005 LICENSE.sax
-rw-r--r--1 root root 2698 Jun 2 2005 LICENSE.xerces
-rw-r--r--1 root root 747 Jun 2 2005 NOTICE
-rw-r--r--1 root root 2657 Jun 2 2005 README
-rw-r--r--1 root root 289 Jun 2 2005 TODO
-rw-r--r--1 root root 18478 Jun 2 2005 welcome.html
-rw-r--r--1 root root 109297 Jun 2 2005 WHATSNEW
```

Como fizemos ao java, iremos setar as variáveis de ambiente do ANT,segue abaixo:

```
[root@nodeB opt]# export ANT_HOME=/opt/apache-ant-1.6.5
[root@nodeB opt]# export PATH=$ANT_HOME/bin:$PATH
[root@nodeB opt]# which ant
/opt/apache-ant-1.6.5/bin/ant
[root@nodeB opt]# ant -version
Apache Ant version 1.6.5 compiled on June 2 2005
```

Apêndice B

Apêndice B - Instalação do Torque (OpenPBS)

B.1 Instalando o PBS

Por padrão o PBS irá usar o rsh ou rcp, ferramentas para copiar arquivos em torno de entrada e saída, mesmo que os jobs estejam rodando em apenas um nó do cluster.

Vamos primeiramente obter o rsh e configurá-lo corretamente. Provavelmente o rsh não estará instalado na distribuição do Fedora. Ele configurado de forma incorreta apresenta risco de segurança para o sistema, porém as configurações seguintes são seguras e testadas.

Como usuário root iremos verificar se o xinetd esta instalado:

```
[root@nodeB ~]# rpm -qa|grep xinetd
xinetd-2.3.13-6
```

Iremos conferir se este está configurado corretamente:

```
[root@nodeB ~]# /sbin/chkconfig --list | grep xinetd
xinetd 0:off 1:off 2:off 3:on 4:on 5:on 6:off
xinetd based services:
```

Se o xinetd não estiver disponível, devemos instalá-lo através do yum

```
[root@nodeB ~]#yum install xinetd
```

Próximo passo sera instalar os pacotes necessários do rsh:

```
[root@nodeB ~]#yum install rsh-server
[root@nodeB ~]#yum install rsh
```

Por padrão o rsh e o rlogin estão desativa-los, vamos verificar entrando nesses dois arquivos. Podemos abrir-los com qualquer editor de texto:

```
/etc/xinetd.d/rsh
/etc/xinetd.d/rlogin
```

E mudar a opção 'disable' para 'no'. Depois de efetuar as instalações e configurações , vamos reiniciar o xinetd:

```
/etc/init.d/xinetd restart
```

Agora iremos iniciar a instalação do torque, para isso vamos baixar o pacote de instalação, podemos baixar neste endereço: http://clusterresources.com/downloads/torque/torque-2.0.0p7.tar.gz

Agora como usuário root vamos executar alguns comandos para descompactar, configurar, construir e por fim instalar o Torque:

```
[root@nodeB ~]#tar -zxf torque-2.0.0p7.tar.gz
[root@nodeB ~]#cd torque-2.0.0p7
[root@nodeB ~]#./configure --prefix=/opt/pbs
[root@nodeB ~]#make
[root@nodeB ~]#make install
[root@nodeB ~]#make packages
[root@nodeB ~]#./torque-package-clients-linux-i686.sh --install --destdir /opt/pbs
[root@nodeB ~]#./torque-package-mom-linux-i686.sh --install --destdir /opt/pbs
```

Note que podemos instalar o PBS em outro diretório para efeitos deste tutorial nos iremos usar como padrão o diretório /opt/pbs.

Como usuário root execute o comando a seguir para iniciar a configuração inicial do PBS server:

```
[root@nodeB ~]#/opt/pbs/sbin/pbs_server -t create
```

Agora vamos executar o comando abaixo:

```
[root@nodeB ~]#/opt/pbs/bin/qmgr
```

Quando qmgr é executado ele irá iniciar um "prompt ", onde precisam ser inseridos certos comandos para efetuar a configuração:

```
Qmgr: set server operators = root@nodb.ufla.br
Qmgr: create queue batch
Qmgr: set queue batch queue_type = Execution
Qmgr: set queue batch started = True
Qmgr: set queue batch enabled = True
```

```
Qmgr: set server default_queue = batch
Qmgr: set server resources_default.nodes = 1
Qmgr: set server scheduling = True
Qmgr: quit
```

Em seguida temos que configurar o PBS, para ele saber quantos e quais nós do cluster estão disponíveis para serem utilizados: Como usuário root iremos criar o arquivo file /usr/spool/PBS/server-priv/nodes

```
[root@nodeB ~]#touch /usr/spool/PBS/server_priv/nodes
```

Com um simples editor de texto, iremos editar o arquivo criado e adicionar linhas para cada maquina do cluster,como o exemplo abaixo :

```
nodeB.ufla.br np=2
nodeA.ufla.br np=2
nodeC.ufla.br np=2
```

Note: Cada linha do arquivo significa uma maquina do cluster, ou seja se no cluster existir 20 maquinas, terão de ser adicionadas 20 linhas, um detalhe para um parâmetro np=2, isso quer dizer o numero de processadores da maquina, no nosso caso as maquinas utilizados tem 2 processadores, mas podem ser que tenham só um ou seja ficarão com np=1.

Vamos dar um comando cat no arquivo para ver como ele fica:

```
[root@nodeB torque-2.0.0p7]# cat /usr/spool/PBS/server_priv/nodes
nodeB.ufla.br np=2
nodeA.ufla.br np=2
nodeC.ufla.br np=2
```

Próximo passo temos que configurar o PBS para que ele saiba qual nó do cluster é a maquina servidora, diferenciando das maquinas 'escravas'. Vamos criar o seguinte arquivo /usr/spool/PBS/mom-priv/jobs/config e edita-lo como o exemplo abaixo:

```
[root@nodeB mom_priv]# cat /usr/spool/PBS/mom_priv/jobs/config
$pbsserver    nodeB.ufla.br
$logevent    255
```

Note: Que no nosso caso a maquina servidora sera o nodB.ufla.br

Como usuário root vamos executar três comandos para iniciar os componentes básicos para o funcionamento do PBS:

```
[root@nodeB ]# /opt/pbs/sbin/pbs_mom
[root@nodeB ]# /opt/pbs/bin/qterm -t quick
[root@nodeB ]# /opt/pbs/sbin/pbs_server
```

Depois de alguns segundos poderemos ver quais nos do cluster estão disponíveis para execução de trabalhos:

```
[root@nodeB mom_priv]# /opt/pbs/bin/pbsnodes -a
nodeB.ufla.br
state = free
np = 2
ntype = cluster
status = opsys=linux,uname=Linux nodeB.ufla.br 2.6.11-1.1369_FC4smp
#1 SMP Thu Jun 2 23:08:39 EDT 2005 i686, sessions=1894, nsessions=1,
nusers=1, idletime=7528, totmem=3065064kb, availmem=3015308kb,
physmem=1033456kb, ncpus=2,loadave=0.00, netload=127730552,
state=free, jobs=? 0, rectime=1139525328
```

Note que somente uma maquina esta habilitada, pois para que as demais fiquem habilitadas temos que iniciar os três serviços nas demais maquinas.

Agora o PBS já esta pronto para aceitar jobs, porem temos que iniciar o serviço de sheduler do PBS ao qual ele poderá executar os jobs em sua fila:

```
/opt/pbs/sbin/pbs_sched
```

Agora, qualquer usuário sobre nodeb poderá submeter jobs a ele através do PBS, e poderá fazer consultas sobre o status do job. Para teste usaremos um usuário genérico 'nemesio' executando os seguintes comandos.

Este comando irá submeter um job simples ao qual o job ficara em modo sleep por 60 segundos e depois retornara a data do sistema:

```
[nemesio@nodeB ~]$ echo "sleep 60;date" | /opt/pbs/bin/qsub
```

Parta saber o status do job, se ele ainda esta sendo executado, ou se sua execução terminou, basta executar o comando abaixo:

```
[nemesio@nodeB ~]$ /opt/pbs/bin/qstat
Job id Name User Time Use S Queue
------------------------------------------------------------
1.nodeb STDIN nemesio 0 R batch
```

Com o job terminado, verificamos a criação de depois arquivos, um arquivo de erro e outro arquivo de saída com o resultado, como não houve nenhum erro o arquivo de erro é criado porem estará vazio, o único conteúdo será o do arquivo de saída.:

```
[nemesio@nodeB ~]$ ls
STDIN.e4 STDIN.o4
```

O conteúdo do arquivo "STDIN.o4"sera o seguinte:

```
[nemesio@nodeB ~]$ cat STDIN.o4
Thu Feb 9 17:09:53 CST 2006
```