UNIVERSIDADE FEDERAL DE SANTA CATARINA

Centro Tecnológico

Departamento de Informática e de Estatística

PORTAIS DE GRIDS COMPUTACIONAIS

Orientador: Profº. Mário Dantas, PhD.

Aluno: Elton Luís Minetto

Florianópolis
Santa Catarina - Brasil
Fevereiro de 2005

Elton Luís Minetto

Portais de Grids Computacionais

"Trabalho de Monografia apresentado ao Departamento de Informática e de Estatística (CTC) área de Ciência da Computação da Universidade Federal de Santa Catarina (UFSC), como requisito para obtenção do Título de Especialista em Ciência da Computação."

Orientador: Profº. Mário Dantas, PhD.

Resumo

Neste trabalho são apresentadas as teorias e tecnologias envolvidas com os Grids Computacionais. Os Grids são considerados como uma solução para um crescente problema enfrentado pelas organizações, que é a necessidade de processar e armazenar uma quantia cada vez maior de dados. Com o advento desta tecnologia as organizações podem utilizar-se de toda a capacidade ociosa e latente de suas máquinas, aplicativos e dispositivos. Este trabalho aborda as características e componentes dos Grids, bem como suas principais vantagens apresentando algumas ferramentas de desenvolvimento mais utilizadas em sua implementação.

Também são descritos os Portais de Grids. Os Portais são apresentados como uma ferramenta para facilitar o acesso por parte dos usuários a todas as características e vantagens providas pelos Grids. Usando-se os Portais, uma união entre a tecnologia dos portais de informação encontrados na Internet e os Grids, os usuários podem facilmente interagir com o ambiente computacional, submetendo procedimentos e acompanhando seu processamento. Além disto, facilitam aos administradores gerenciar os recursos providos pelo grid para tirar o melhor proveito possível de suas capacidades. São apresentadas também algumas ferramentas encontradas no mercado para a implementação dos portais.

Abstract

In this work some theories and technologies involved with Computational Grids are presented. The Grids are considered to be a solution for a growing problem faced by organizations which is the need to process and store data amount that becomes bigger each time. With this technology advent the companies can take advantage of all the idle and latent capacity from their machines, softwares and devices. This work approaches Grids' features and components, as well as the main advantages, presenting some development's tools most used in its implementation.

The Grids Portals are also described. The Portals are presented as a tool to make the access easier for the users in all the advantages and features provided by the Grids. By using the Portals, an union between the information portals' technologies found on the internet and the Grids, the users can easily interact with the computational environment, submitting procedures and accompaning its processing. Besides it, they make it easier for the administrators to manager the resources provided by the grid to get the best possible profit of their capacities. Some tools found in market for the portals' implementation are also presented.

Índice Geral

Lista de Figuras

Lista de Abreviaturas

API	Application Program Interface
CA	Certificate Authority
CGI	Common Gateway Interface
CoG	Commodity Grid
DNS	Domain Name Service
FTP	File Transfer Protocol
GIIS	Grid Index Information Service
GGF	Global Grid Forum
GRAM	Grid Resource Allocation and Management
GRIS	Grid Resource Information Service
GSI	Grid Security Infrastructure
HTML	HyperText Markup Language
ICMP	Internet Control Message Protocol
LDAP	Lightweight Directory Access Protocol
MDS	Metacomputing Directory Service
MPI	Message Passing Interface
OGSA	Open Grid Services Architeture
OSPF	Open Shortest Path First
OV	Organização Virtual
RSL	Resource Specification Language
SSL	Secure Socket Layer
SOAP	Simple Object Access Protocol
TCP/IP	Transmission Control Protocol/Internet Protocol
UDP	User Datagram Protocol
XML	Extensible Markup Language
WSDL	Web Services Definition Language

1. Considerações Iniciais

1.1 Introdução

É fato que as instituições e empresas tem, com o passar dos anos, acumulado uma quantia muito grande de dados e informações. Dados de produtos produzidos e vendidos, contatos com clientes, informações de mercado, resultados de pesquisas científicas ou de marketing. Com o advento das redes e principalmente da Internet, adquirir informação não é mais um problema, pois ela está disponível em quantidades gigantescas. O principal problema hoje, é tirar o melhor proveito possível desta informação acumulada e processá-la de maneira a produzir resultados. Para isso, a capacidade de armazenamento e principalmente de processamento destes dados precisa corresponder a este aumento.

Outro fato importante a ser considerado é que o hardware teve uma redução de seus custos nos últimos anos. Hoje adquirir um computador comum para desktop chega a custar apenas algumas centenas de dólares, diferente dos milhares que custava alguns anos atrás.

Unindo-se estes dois fatores, temos uma combinação importante. As organizações precisam processar quantidades enormes de dados e possuem em seu parque de máquinas dezenas ou centenas de computadores que passam a maior parte do tempo ociosas, seja por estarem desligadas ou por serem usadas apenas como ferramentas de automação de escritórios.

A tecnologia dos Grids Computacionais usa este cenário para prover uma solução a este problema. A idéia principal dos Grids é usar a capacidade ociosa dos computadores para processar dados ou armazená-los. Usando-se a tecnologia dos Grids uma organização pode utilizar toda a capacidade de seu parque de máquinas em seu proveito, unindo desde aplicações, servidores, bases de dados, armazenamento, capacidade de processamento, tratando tudo como um serviço virtualizado. Desta forma, usando os recursos computacionais de maneira confiável e transparente, não se importando em qual computador foi processado ou está armazenado a informação que precisa.

Mas as vantagens que os Grids fornecem não seriam utilizados em toda sua plenitude se o acesso a eles for complexo e difícil. O usuário necessita de uma forma de interagir com o Grid de maneira simples, agradável e produtiva. Se ele precisar executar diversos comandos ao iniciar uma aplicação ou procedimento no Grid, uma grande barreira será levantada. Para

melhorar o acesso por parte dos usuário e o gerenciamento feito pelos administradores, tem sido desenvolvidos e usados os Portais de Grids. Os portais unem as características e facilidades que a Internet trouxe às empresas com a tecnologia dos Grids. Através de uma interface amigável os usuários podem submeter novos procedimentos ao Grid ou acompanhar o processamento dos já em execução. Um administrador pode visualizar de maneira rápida e completa o estado do Grid, quantos usuários estão utilizando, qual é a capacidade fornecida e usada, se é necessário expandir, entre outras características.

Nos próximos capítulos são apresentados detalhes aprofundados sobre os Grids, suas características e vantagens, seus componentes principais e algumas ferramentas existentes. Também são apresentados os Portais de Grids, suas características e algumas soluções existentes hoje, tanto comerciais quanto acadêmicas.

2. Grids

2.1 Introdução

O principal conceito que pode ser atribuído aos Grids é o de usuários separados geograficamente compartilhando, de forma dinâmica, recursos computacionais (Miley, 2003). Pode ser considerado uma evolução da computação distribuída. A principal idéia é criar a ilusão de um computador virtual, em grande escala, facilmente gerenciável e com uma vasta quantidade de poder computacional e dispositivos sendo compartilhados (Berstis, 2003). Sendo assim, Grids removem as conexões fixas entre aplicações, servidores, bases de dados, máquinas, armazenamento, entre outros, tratando tudo como um serviço virtualizado (Nash, 2004). Os dispositivos podem estar em uma mesma sala ou distribuídos através do mundo, podem estar utilizando diferentes plataformas de hardware ou diferentes sistemas operacionais, podendo pertencer a diferentes empresas ou instituições.

Outra idéia bastante defendida é que a computação seja confiável e transparente como uma utilidade. 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 á utilização da energia elétrica. Quando um interruptor é acionado 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 (Pitanga, 2003).

Estes conceitos não são novos. De acordo com Foster *(apud* Miley, 2003, p. 40), estas idéias tem sido levantadas desde 1965. No MIT[1], onde pesquisadores e desenvolvedores responsáveis pela criação do sistema operacional Multics, liderados por Fernando Corbató, visualizavam a hipótese de dispositivos computacionais compartilhando seus recursos e poder de processamento semelhantemente a uma rede elétrica ou de água.

Da mesma maneira que a Internet, a computação em Grid, teve seu desenvolvimento nas comunidades acadêmica e de pesquisa. Entre os fatores que influenciaram seu desenvolvimento podem ser considerados importantes:

- devido ao fato de pesquisas incluírem um grande número de pesquisadores e estes, geralmente, encontrarem-se em locais dispersos, notou-se a necessidade de criar um

1 *Massachusetts Institute of Techonoly* – Importante instituição de pesquisas americana

ambiente computacional para compartilhar recursos e resultados dinamicamente;

- também era importante escalar facilmente, para poder acomodar uma quantia cada vez maior de dados e poder computacional;
- e, manter os custos baixos.

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

Deste modo, sua utilização no meio acadêmico foi de grande importância e de grande utilidade em diversos projetos. O projeto SETI@home, *Search for Extraterrestrial Intelligence*, foi um dos maiores exemplos de Grids no meio científico. Sinais de telescópios, receptores de rádio e outras fontes de monitoramento eram distribuídos para computadores através da Internet. Esses computadores processavam os dados em busca de sinais que pudessem comprovar a existência de vida extraterrestre, utilizando seu tempo ocioso de processamento.

Para esclarecer melhor a definição, Ian Foster (*apud* Bombonato, 2003) definiu três características básicas para que um sistema computacional possa ser chamado de Grid:

- 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. Pode haver um controle local em uma empresa, mas não existe um controle central para todo o grid.
- Utilizar padrões abertos, interfaces e protocolos de propósito geral. A utilização de protocolos e padrões abertos é essencial para que os sistemas em Grid possam realizar funções fundamentais como autenticação, autorização, descobrimento de recursos e acesso a eles, sem perder a capacidade de escalar e interagir com diferentes plataformas de hardware e software.
- Prover o mínimo em qualidade de serviços, como segurança, tempo de resposta e disponibilidade.

Em Bombonato (2003), Ian Baird cita que a adoção em massa da computação em grid deve acontecer em três fases:

- Primeira Fase: Enterprise Grids. É o início, com empresas que possuam presença em diversas cidades, países ou sedes, compartilhando recursos, tudo isso restrito ao domínio da organização, atrás dos limites do firewall. Desta forma múltiplos projetos

ou departamentos dentro da organização podem compartilhar seus recursos computacionais, utilizando-os de forma otimizada em tarefas como engenharia colaborativa, mineração em grandes bases de dados e renderização de frames para animação.

- Segunda Fase: Partner Grids. Uma expansão e interligação entre grids de organizações de áreas de interesse e pesquisa semelhante. Empresas que trabalham no ramo farmacêutico, por exemplo, tornando-se parceiras e compartilhando recursos para atingir um objetivo em comum.
- Terceira Fase: Service Grids. Os sistemas em grid tornar-se-ão um sistema utilitário, com usuários utilizando recursos sem saber onde estão localizados ou a que organização pertencem, apenas pagando pelo uso destes.

2.2 Benefícios

O conceito de Grids computacionais está se tornando uma grande tendência no atual momento. Grandes desenvolvedores de software e de hardware estão prevendo que essa pode ser uma das grandes possibilidades de evolução da computação. Como em toda tecnologia em ascensão, cria-se uma grande expectativa em relação ao que pode-se ou não fazer com ela. Como grandes capacidades dos grids Berstis (2003) cita:

a) Explorar recursos subutilizados e recursos adicionais

Basicamente, uma aplicação pode ser executada em qualquer das máquinas participantes, desde que estas possuam acesso a determinados recursos solicitados pela aplicação. Pode-se escolher esta máquina utilizando-se diversos parâmetros, como a que estiver com menor carga de processamento, a que possuir disponível certo tipo de dispositivo ou determinados dados. Existem alguns tipos de aplicação que podem melhor utilizar essas características dos Grids. Aplicações que exigem grande processamento de dados e pouca interação com o usuário podem ser melhor escalonadas através do Grid.

Além dos recursos de processamento muitas máquinas também possuem seus discos rígidos sendo subutilizados. Assim, o Grid pode ser utilizado como um *Data Grid* alocando o espaço disponível como se fosse um disco apenas. Outra forma de alocar o espaço seria dividir os dados de forma que as aplicações possam ser executadas em uma máquina mais

próxima de onde se encontram os dados que processa, ou para garantir uma maior disponibilidade caso alguma máquina falhe.

Diversos outros recursos podem ser compartilhados em um grid. Para uma aplicação que demande de um maior acesso à Internet, por exemplo, pode-se dividir o trabalho entre outras máquinas que também possuam acesso à rede, acelerando os resultados. Outros exemplos podem abranger uma impressora remota com maior qualidade, um gravador de DVD ou equipamentos médicos e científicos avançados como um microscópio eletrônico ou um robô.

Com isso, os recursos de uma instituição ou empresa podem ser melhor utilizados, diminuindo despesas e aumentando a eficiência e a competitividade.

b) Capacidade de processamento paralelo

Outra característica interessante é a possibilidade de melhor utilizar o processamento paralelo através de Grids. Em alguns tipos de aplicações como nas científicas, financeiras, processamento de imagens e simulações, a utilização de processamento paralelo pode agilizar e muito seu trabalho.

Uma aplicação escrita utilizando-se de algoritmos e técnicas de programação paralela pode ser dividida em partes menores e estes podem ser separados e processados independentemente. Cada uma destas partes de código podem ser executadas em uma máquina distinta no Grid, melhorando a performance.

Existem algumas barreiras que podem impedir que uma aplicação utilize todo este potencial. Por exemplo, se a aplicação pode ser dividida em um número fixo de partes independentes, isso torna-se uma barreira que impede sua escalabilidade. Outra forma de problema encontrado é quando as partes não podem ser completamente independentes e precisam comunicar-se entre si, causando uma possível espera para que as comunicações sejam sincronizadas ou o tempo necessário para as mensagens serem transmitidas. Além disso, as partes podem precisar acessar uma base de dados ou um outro tipo de recurso. Essas características devem ser levadas em conta no momento de se utilizar à funcionalidade de processamento paralelo, mas isso não impede a grande utilização dos Grids como uma excelente arquitetura para o processamento paralelo.

c) 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 chamado 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 (Pitanga, 2003). Pode ser definida de acordo com sua área de pesquisa ou de negócio, juntando usuários e aplicações com propósitos semelhantes, colaborando em prol de uma razão comum, como por exemplo, empresas, centros de pesquisa e universidades. Esses recursos podem abranger processamento, dados, licenças, entre outros, e podem ser "virtualizados" para melhor interoperabilidade entre os participantes do Grid.

d) 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.

Utilizando-se uma abordagem baseada em Grids, com máquinas espalhadas em diversos lugares diferentes, quando uma falha atinge uma parte do Grid as demais podem continuar sua operação normalmente. Sistemas de gerenciamento podem executar novamente processos importantes caso seja detectada alguma falha ou estes podem ser executados redundantemente para garantir sua consistência. Dados podem ser duplicados ou separados em diversas partes através do grid, aumentando sua disponibilidade. Um grande avanço nessa área serão sistemas que podem automaticamente detectar uma falha e tomar as medidas necessárias para contornar o problema.

2.3 Componentes

2.3.1 Tipos de Recursos e Grids

Um grid é uma coleção de máquinas compartilhando diversos tipos de recursos. Entre os principais tipos de recursos possíveis de compartilhamento pode-se destacar alguns:

- Computação: Ciclos de processamento são os recursos mais comumente compartilhados em um grid. Os processadores podem variar em velocidade, arquitetura, plataforma de software e ter outros fatores associados como memória e armazenamento. Existem três formas de explorar os recursos computacionais em um grid. O primeiro é executar uma aplicação em qualquer máquina disponível do grid, independentemente de onde esteja localizada. A segunda é quebrar o aplicativo em partes menores para que estas possam ser executadas paralelamente através do grid. E a terceira é executar uma tarefa que precisa rodar várias vezes em diferentes máquinas do grid. Grids que utilizam principalmente os recursos computacionais disponíveis são comumente chamados de *grids computacionais.*
- Armazenamento. A utilização do espaço de armazenamento de cada máquina pelo grid é uma forma interessante de compartilhamento deste recurso. Este espaço pode ser utilizado como se fosse uma memória cache do grid ou como se fosse um sistema de arquivos só. Desta forma aumenta a capacidade de armazenamento como um todo, além de aumentar a performance, compartilhamento e confiabilidade dos dados. Utilizando todo o espaço como se fosse um sistema de arquivos só para todo o grid facilita a localização de determinado arquivo, sendo que este pode estar dividido em partes menores e espalhado pelas máquinas participantes, contornando problemas como tamanho máximo de arquivo em algum sistema operacional. Sistemas de gerenciamento podem duplicar dados sensíveis em várias máquinas provendo uma redundância. Podem também utilizar essa informação no momento de designar uma máquina para executar uma aplicação, escolhendo a que estiver com os dados requisitados por esta. Grids que utilizam o espaço de armazenamento como principal recurso compartilhado são comumente chamados de *data grids* ou *grids de dados.*
- 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 processar um grande número de dados e estes podem não se encontrar na máquina onde está sendo executada. Neste caso os dados devem ser enviados através do grid junto com o executável, sendo que a velocidade

desta transmissão pode influenciar a performance. Então, máquinas com conexões ociosas podem ser utilizadas para enviar partes destes dados para não causar uma contenção. O mesmo pode ser aplicado quando os dados a serem enviados são para fora do grid, como a Internet ou outro tipo de rede conectada. Grids que tem por finalidade principal fornecer alta performance ou redundância no quesito comunicação são conhecidos como *network grids* ou *delivery grids*.

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

2.3.2 Arquitetura

Em Foster (2003a) define-se a arquitetura de um Grid na forma de uma pilha de protocolos, semelhante à pilha de protocolos TCP/IP, como demonstra a Figura 1.

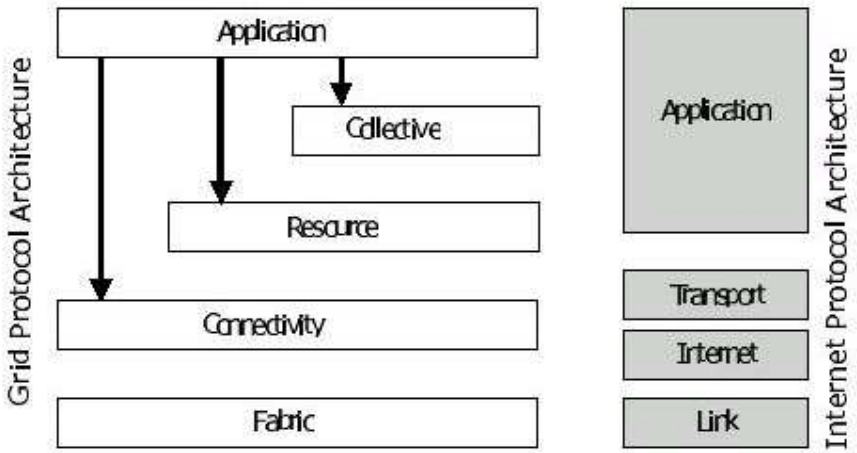

Figura 1 - Arquitetura de um Grid comparado com a arquitetura TCP/IP (Foster 2003a)

a) 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.

Conceitualmente, os recursos precisam ter implementados no mínimo dois

mecanismos importantes para seu melhor aproveitamento no Grid. O primeiro é um mecanismo chamado *enquiry*, que permite descobrir sua estrutura, estado e capacidades. E, um mecanismo de gerenciamento, com o qual é possível algum tipo de controle sobre a qualidade do serviço fornecido.

Como citado, para cada tipo de recurso possível de ser compartilhado através do grid, a camada Fabric implementa algumas funções específicas:

- Recursos computacionais: mecanismos para iniciar programas e monitorar sua execução são necessários. Além disso, é importante possibilitar gerenciamento sobre os recursos alocados para determinada aplicação, e fornecer informações sobre características de hardware, software e carga de processamento.
- Recursos de armazenamento: basicamente são necessárias funções que permitam adicionar e retirar arquivos. Também é necessária a existência de mecanismos de gerenciamento que permitam controle sobre os recursos alocados para a transferência de dados, como espaço, taxa de transferência do disco e da rede, CPU, etc. Fornecer informações sobre as características de hardware, software, espaço em disco disponível e taxa de transferência.
- Recursos de rede: gerenciamento sobre os recursos alocados para transferências de rede, como por exemplo, a taxa de transferência. Também fornecer informações sobre as características da rede e sua carga.

b) Connectivity: Comunicação fácil e segura

A camada Connectivity define os protocolos de comunicação e de segurança necessários para transferências de rede específicas dos grids, permitindo transferência de dados entre recursos, utilizando as funcionalidades da camada Fabric. Fornece também protocolos de autenticação e serviços de segurança criptografada para verificar a identidade de usuários e recursos.

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).

As soluções de autenticação devem possuir no mínimo algumas características básicas. São elas:

- *Single sign on*: os usuários devem se autenticar uma só vez e ter acesso a todos os

recursos definidos pela camada Fabric, não necessitando se autenticar em cada um deles.

- Delegação: um programa executado por um usuário tem acesso aos mesmos recursos que este usuário está autorizado a utilizar. E, um programa pode opcionalmente delegar alguns direitos a outros programas em execução.
- Integração com sistemas de segurança locais: cada recurso pode implementar soluções de segurança localmente, como uma máquina Linux. Os sistemas de segurança do grid devem poder interoperar com estes sistemas.

c) 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. Utiliza também a camada Fabric para acessar e controlar dispositivos locais.

Existem apenas dois tipos de protocolos que são definidos por esta camada:

- Protocolos de informação: usados para obter informações sobre a estrutura e o estado dos recursos, como configuração, carga de trabalho atual e política de uso.
- Protocolos de gerenciamento: usados para negociar acesso a um recurso compartilhado, como por exemplo, os requisitos do recurso e as operações necessárias para a criação de um processo ou acesso a um dado. Podem também monitorar o estado de uma operação e ter controle sobre ela.

d) 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:

- Serviços de diretório: facilita com que organizações virtuais possam descobrir a existência e propriedades de recursos compartilhados. Estes recursos podem ser localizados por nome e outros atributos como tipo, disponibilidade e carga.
- Serviços de alocação e agendamento: permitem que recursos possam ser alocados para

determinado propósito e o agendamento de tarefas em recursos apropriados.

- Serviços de monitoramento e diagnóstico: recursos podem ser monitorados em relação à falhas, utilização indevida e sobrecarga, por exemplo.
- Serviços de replicação de dados: suporte a gerenciamento dos recursos de armazenamento para maximizar a performance de acesso a dados. Pode-se monitorar tempo de resposta, confiabilidade e custos.
- Sistemas de programação compatíveis com Grids: habilita a utilização de modelos de programação conhecidos no ambiente de grids, usando serviços como busca e alocação de recursos, segurança, etc. Um exemplo seria a implementação de Message Passing Interface (MPI) no ambiente de grid.

Alguns componentes desta camada podem ser específicos para determinada organização virtual ou domínio de aplicações. Por exemplo, um protocolo para alocação de um tipo específico de dispositivos de rede que esta organização possui. Outros componentes podem ser mais generalizados, como um serviço de replicação que gerencia uma coleção internacional de dispositivos de armazenamento para diversas organizações virtuais. De qualquer forma, o importante é que os protocolos e interfaces de programação sejam baseados em padrões abertos para evitar problemas de integração.

e) Applications

A camada final da arquitetura grid compreende as aplicações de usuário que operam no ambiente de um grid. A Figura 2 demonstra a arquitetura de grid pelo ponto de vista de um programador de aplicativos. 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.

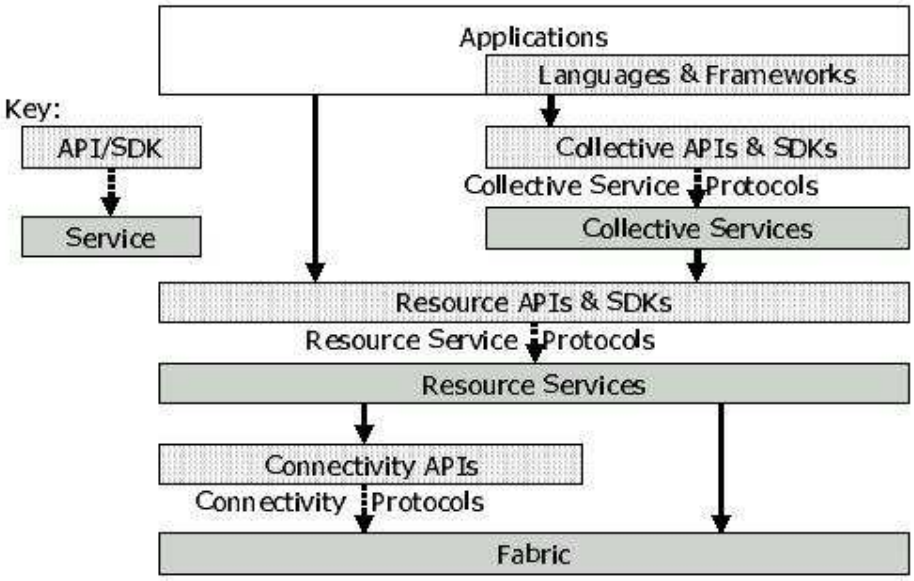

Figura 2 - Arquitetura de Grid na visão da programação (Foster 2003a)

2.4 Ferramentas de Desenvolvimento

Existem algumas ferramentas disponíveis hoje para o desenvolvimento de aplicativos que façam uso da arquitetura de grids. Além de implementações comerciais, mantidas por algumas empresas, dois projetos têm alcançado grande popularidade, principalmente por serem desenvolvidos a partir de padrões abertos e mantidos por comunidades de programadores. São o projeto Globus Toolkit e o Legion.

2.5.1 Globus Toolkit

O Globus Toolkit é um conjunto de ferramentas e bibliotecas de software que dão suporte a arquitetura e as aplicações em Grid. É um projeto desenvolvido pelo *Argnone National Laboratory* (ANL) e *University of Southern California.* É mantido por uma comunidade de programadores e é baseada em arquiteturas e códigos abertos. Com ele é possível implementar segurança, busca de informações, gerenciamento de recursos e de dados, comunicação, detecção de falhas e portabilidade (Dantas, 2003).

Segundo Foster (2003b), os serviços no globus são baseados em um modelo conhecido como *Hourglass* (Ampulheta). Nesta analogia, os serviços locais encontram-se na parte inferior da ampulheta, enquanto que os serviços globais estão na parte superior. Os

serviços fornecidos pelo globus ficam localizados no gargalo da ampulheta.

Os componentes do conjunto de ferramentas Globus podem ser usados tanto independentemente quanto em conjunto no desenvolvimento de aplicações e de ferramentas de programação de grid. Para cada componente existe uma API[2] em linguagem C ou Java definida para facilitar o uso pelos desenvolvedores.

A Figura 3 demonstra a estrutura do Globus Toolkit.

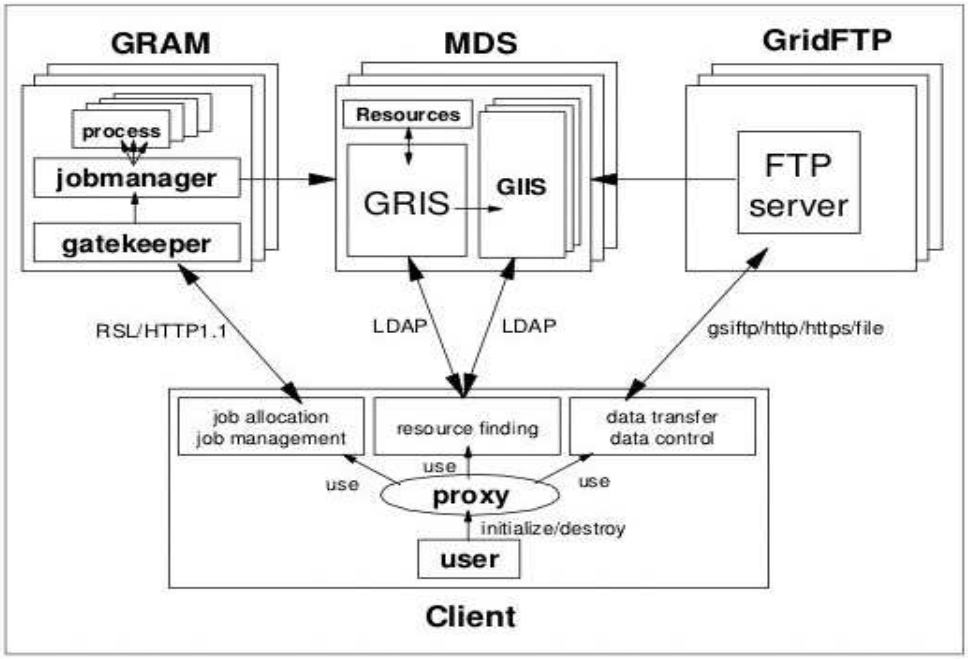

Figura 3 - Estrutura do Globus Toolkit (Ferreira 2002)

Seus componentes mais importantes, segundo Foster (2003c) e Ferreira (2003), são:

a) *Grid Resource Allocation and Management* (GRAM)

Age como uma interface entre serviços locais e globais, traduzindo requisições de recursos genéricos em comandos que são específicos para um sistema local particular. Um grid construído sobre as ferramentas globus pode conter vários GRAMs, sendo cada um responsável por um conjunto local de recursos em particular (Pitanga, 2003).

O software *gatekeeper* pode ser considerado como uma interface entre o usuário e o GRAM, pois é responsável pela autenticação do usuário e sua associação com uma conta no computador local. Também prove segurança, confiabilidade, criação e gerenciamento de

2 *Application Programming Interface*

serviços.

Outra função do *gatekeeper* é iniciar o processo *job manager* após ter autenticado o usuário. O *job manager*, por sua vez, é responsável pela comunicação com usuários locais, alocação de recursos, execução de tarefas e por desalocar os recursos após a conclusão da tarefa (Pitanga, 2003).

O GRAM também conta com uma linguagem utilizada para comunicar requisições de recursos. Conhecida como *Globus Resource Specification Language* (RSL), é uma linguagem extensível e simples, composta por uma lista de expressões de recursos logicamente combinados com alguns operadores lógicos, como demonstra o exemplo abaixo:

```
&
(jobtype = mpi)
(executable = /home/elm/grid/exemplo)
(arguments = "-v")
(count = 1)
(stdout = /home/elm/grid/exemplo.out)
(stderr = /home/elm/grid/exemplo.err)
(environment = (GRD_PE mpi-hpc01))
```

b) Metacomputing Directory Service (MDS)

Prove um conjunto de ferramentas para descobrir e acessar configurações e informações de sistemas, da sua utilização, do gerenciamento e a respeito da rede. Como exemplo de informação, pode-se citar carga da rede, a situação de um recurso, processos sendo executados, arquitetura e sistema operacional de determinada máquina, carga do sistema, entre outros. É baseado em LDAP[3] e composto de duas partes, o *Grid Resource Information Service* (GRIS) e o *Grid Index Information Service* (GIIS).

c) Grid Security Infrastructure (GSI)

Prove serviço de autenticação do tipo *single sign on*, com suporte a controle local, delegação e mapeamento de credenciais. É usado para autenticação, autorização e delegação de credenciais para computações remotas.

Emprega o protocolo de autenticação ITU X.509 [Pit03], que especifica a utilização de certificados. A segurança baseada em certificados, credenciais que provêem identidade, conta exclusivamente com a confiança da *Certificate Authority* (CA), que prove a cada entidade seu próprio certificado (Pitanga, 2003).

3 LDAP (*Lightweight Directory Access Protocol*) é um protocolo de serviço de diretório usado sobre TCP/IP.

Após a autenticação ter sido realizada a comunicação do globus é desempenhada utilizando o protocolo *Secure Socket Layer*[4] (SSL), que cifra as mensagens.

d) GridFTP

É um componente chave para a transferência de arquivos de modo seguro e com alta performance. A palavra GridFTP pode se referir a um protocolo, um servidor ou a um conjunto de ferramentas. O protocolo é baseado no protocolo FTP[5], mas o extende de modo que suporte as características da arquitetura de grid e outras ferramentas do Globus como o GSI.

e) Biblioteca Nexus

É uma biblioteca responsável pelos serviços de comunicação no conjunto de ferramentas globus. Esta define uma interface de programação para suportar alguns paradigmas de programação importantes, como passagem de mensagens, chamadas de procedimento remoto, entre outras (Dantas, 2003).

2.5.1.1 Globus Toolkit 3

O Globus Toolkit 3 (GT3) apresenta uma evolução em relação a suas versões anteriores. Ele é baseado no conceito de Grid Services, uma expansão do conceito de Web Services. Os principais conceitos e padrões nos quais está baseado o GT3 serão discutidos nas seções seguintes.

a) Open Grid Services Architecture (OGSA): como citado, a utilização de padrões é uma das principais necessidades para que os sistemas em grid sejam utilizados amplamente. A arquitetura da computação em grid é definida pela *Open Grid Services Architeture* (OGSA), desenvolvida pelo *Global Grid Forum* (GGF). OGSA define o que são os serviços e toda a estrutura que pode ser provida em um ambiente grid . É baseada nos padrões já definidos para os Web Services e considera um serviço em um grid como um Web Service com algumas particularidades definidas através da linguagem padrão chamada WSDL (*Web Services*

4 SSL (*Secure Socket Layer*) protocolo desenvolvido pela Netscape Communications com o objetivo de gerar segurança e privacidade entre duas aplicações, utilizando criptografia.
5 FTP (*File Transfer Protocol*) protocolo de transferência de arquivos utilizado na Internet

Definition Language), com pequenas extensões. Isto é importante porque fornece uma maneira de acessar um serviço em grid usando padrões abertos e consolidados, como SOAP[6], XML[7] e WS-Security[8]. Além disso, padronizam a pesquisa, identificação e utilização de novos serviços conforme forem sendo adicionados ao grid.

b) Open Grid Services Infrastructure (OGSI): OGSI é a especificação concreta da infraestrutura da OGSA. Baseado nas tecnologias de Grids e Web Services, é o *middleware* para os chamados *grid services*, ou serviços do Grid, definindo como construir, gerenciar e expandir um serviço.

c) Web Services: os web services são a base para o conceito de Grid Services e, por sua vez, a base para o OGSI, OGSA e o GT3. Web Services é uma tecnologia de computação distribuída que permite a criação de aplicações cliente/servidor usando como plataforma de comunicação protocolos abertos e amplamente conhecidos como o HTTP e o XML. Desta forma, pode-se executar um serviço através de uma intranet ou mesmo Internet, ocultando do cliente detalhes relativos a implementação (se o serviço foi desenvolvido em Java ou Python, por exemplo) e localização (se está sendo executado em um servidor Linux, Windows ou em um mainframe), bastando ao cliente conhecer a URL pela qual o mesmo pode ser acessado e os tipos de dados de seus métodos. Os Web Services são definidos pelo World Wide Web Consortium (W3C), o mesmo grupo que define os padrões da Internet de modo geral. O W3C definiu uma maneira padronizada para descrever os web services, o WSDL. Basicamente este padrão define como são descritos os métodos, parâmetros, tipos de dados, protocolo de transporte e URI (Uniform Resource Identifier) de um serviço. Para um web service ser acessado pelos clientes o mesmo precisa ser publicado em um servidor. Deve também existir uma forma com a qual possa ser realizado pesquisas em busca de um determinado serviço. Para suprir estas necessidades, o W3C definiu um padrão chamado UDDI (Universal Description, Discover and Integration). Um protocolo padrão para comunicação também foi definido, o SOAP (Simple Object Access Protocol), sendo baseado em XML e usando o HTTP como transporte.

A Figura 4 mostra o processo de invocação de um web service (Ferreira, 2004):

1. O cliente utiliza o Registro UDDI para descobrir onde o serviço desejado está sendo fornecido.
2. O Registro UDDI responde ao cliente com o endereço do serviço na forma de um URI que

6 SOAP (*Simple Object Access* Protocol) é um padrão de comunicação com web services.
7 XML (*Extensible Markup Language*) é uma linguagem de marcação de dados que provê um formato para descrever dados estruturados.
8 WS-Security é uma especificação que suporta, integra e unifica vários modelos, mecanismos e tecnologias de segurança em uso no mercado, permitindo que vários sistemas possam interoperar em plataformas e linguagens neutras.

aponta para o servidor que fornece o serviço desejado. O URI é semelhante a uma URL de uma página Web. Esta semelhança vem do fato de que geralmente um serviço web está armazenado em um container web. Um exemplo de URI: http://www.webservice.server.com/application/servico

3. Com esta informação o cliente conhece a localização do serviço, mas ainda não sabe como invocá-lo. Então, o cliente pergunta ao servidor como pode invocar o serviço desejado.
4. O servidor responde ao cliente enviando um documento WSDL, que descreve a interface do serviço, ou seja, seus métodos, parâmetros e tipos de dados.
5. De posse destas informações, o cliente sabe como invocar o serviço, sendo que esta invocação pode ser realizada usando-se diversos protocolos. O protocolo padrão para esta função é o SOAP, o cliente então faz a chamada ao serviço usando o mesmo.
6. O serviço responde então com uma resposta SOAP, encapsulando os dados na gramática XML, conforme definido pelo W3C.

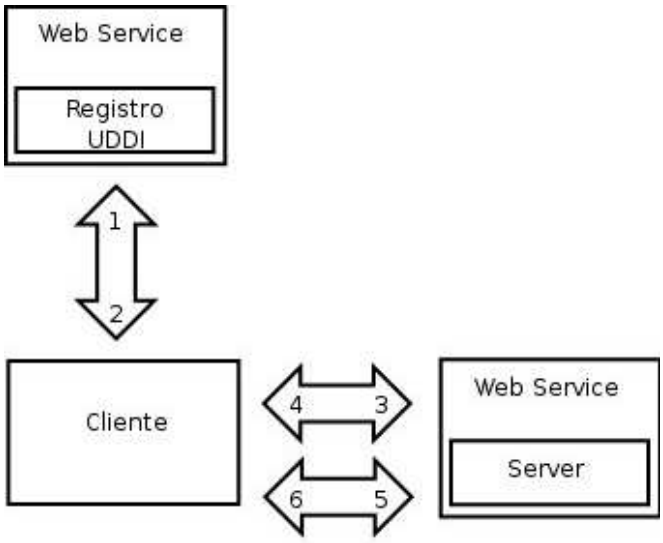

Figura 4 - Processo de invocação de web service(Ferreira 2004)

d) ***Grid Services:*** os Grid Services são baseados no conceito de Web Services com algumas particularidades e mecanismos definidos pelo OGSI. Os principais conceitos referentes aos grid services, segundo Ferreira, 2004, são:

Nomeação (naming) : da mesma maneira que um web service possui uma URI associada a ele, o grid service possui uma identificação, conhecida como *Grid Service Handle* (GSH). Assim como uma URI é única de modo a identificar apenas

um web service, o GSH também deve identificar um único grid service. O GSH não fornece informações detalhadas sobre o grid service, apenas seu endereço. Desta forma, para conhecer os detalhes e requisitos de comunicação com o serviço, uma GSH é mapeada a um *Grid Service Reference* (GSR). Então, um GSH aponta para um grid service e o GSR especifica como deve ser a comunicação com este serviço. Como o GT3 usa a tecnologia de web services, SOAP é usado como meio de comunicação entre os serviço e um documento WSDL é usado para a descrição necessária pelo GSR. Resumindo, o GSR nada mais é do que um documento WSDL entregue usando-se SOAP.

Dados do serviço : uma instância de um grid service pode possuir uma coleção de dados estruturados associados a ela e que podem ser acessados pelos usuários do serviço. Estes dados, chamados de *Service Data Elements* (SDE) podem ser pesquisados e alterados pelos usuários. A especificação OGSI define alguns tipos padrão de SDE que podem ser usados pelos grid services, sendo que um grid service geralmente possui alguns destes SDEs quando é instanciado.

Notificações: o mecanismo de notificação permite que um serviço envie mensagens para outro, mediante uma inscrição por parte do destino.

Ciclo de Vida: a especificação OGSI define o ciclo de vida de um serviço como o tempo desde a criação de uma instância até a sua destruição. A criação da instância pode ser realizada pelo cliente ao requisitar ao mecanismo Factory, também definido na especificação OGSI, que crie o novo serviço. O processo de destruição pode ser feito ao invocar-se o método da instância responsável por esta tarefa. Outra forma do ciclo de vida é quando o cliente instancia o serviço por um tempo determinado. Quando este período expira, se o cliente não reafirmar o interesse da utilização, o serviço é terminado.

Uma vez discutidos os principais conceitos nos quais se baseia a implementação do GT3, pode-se abordar seus principais componentes, como demonstra a Figura 5. As partes em cinza da Figura 5 representam os componentes que formam o núcleo do GT3.

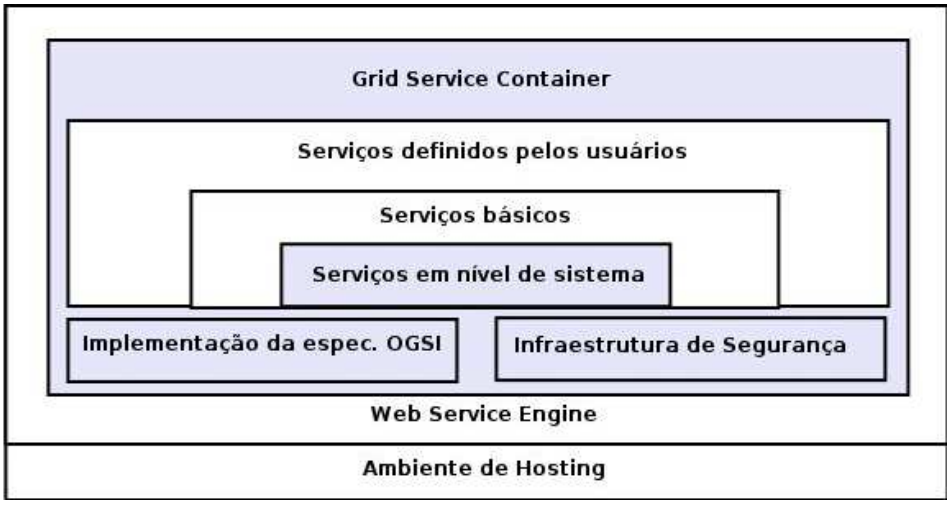

Figura 5 - Componentes do GT3 (Ferreira 2004)

Os componentes mais importantes apresentados na Figura 5 são, segundo Ferreira, 2004:

a) Ambiente de Hosting, Web Service Engine e Grid Service Container: o ambiente de hosting é onde o GT3 é executado. Alguns tipos de ambiente podem são compatíveis com a atual implementação do GT3. Estes ambientes são associados com o Grid Service Container, que por sua vez, tem como alicerces a implementação da especificação OGSI e a infraestrutura de segurança. Os ambientes são escolhidos no momento da instalação ou configuração do GT3 e são:

Embedded Container: utiliza o ambiente embutido no próprio GT3, sendo usado apenas quando clientes precisam criar e gerenciar pequenas quantidades de grid services ou em servidores com pouco tráfego, devido a limitações óbvias quanto a performance.

Stand-alone Container: este modelo de ambiente é uma pequena evolução do ambiente anterior, sendo basicamente o embedded container contando com uma interface ou linha de comando que permite ao usuário iniciar e parar o ambiente. Por exemplo, os comandos globus-start-container e globus-stop-container.

J2EE Web Container (Servlets): este modelo é basicamente a utilização do container sendo executado dentro de uma máquina Java (um Web container). Como exemplo de web container pode-se citar o Tomcat e o IBM WebSphere Application Server entre outros. Neste caso, é utilizado os recursos de web services providos pelo servidor de aplicações e não os fornecidos pelo GT3, aumentando a disponibilidade e performance.

J2EE Enterprise JavaBeans Container(EJB): este modelo é semelhante ao anterior,

mas neste caso executa-se o container dentro de um servidor de aplicações EJB (EJB container) como o IBM WebSphere Application Server ou o Jboss.

b) Serviços em nível de sistema: são serviços básicos fornecidos pelo GT3 para facilitar o uso e administração de grid services em ambientes de produção. Dentre estes serviços podemos citar serviços de administração, de registro de atividades (*logging*), e de gerenciamento de performance e utilização.

c) ***Serviços básicos:*** estes serviços não são parte do núcleo do GT3 sendo necessário sua explicação separada. São muito úteis na utilização do GT3 em ambientes de produção. São eles:

Serviços de gerenciamento de tarefas (job management): provêem métodos com os quais é possível a criação e gerenciamento de tarefas remotamente. Estes métodos invocam um componente chamado de Master Managed Job Factory Service (MMJFS).

Serviços de indexação (index services): usados principalmente para fins de busca de serviços. Fornece comandos com os quais um cliente pode realizar pesquisas por qualquer SDE de qualquer grid service.

Serviços de transferência de arquivos confiável (Reliable File Transfer): RFT ou multiRFT é parte da implementação Data Management, assim como o GridFTP e o Replica Relocation Service (RLS). Com estes serviços é possível a transferência de arquivos de maneira confiável dentro do grid. Na parte cliente é fornecido uma interface desenvolvida em Java, enquanto que no servidor é utilizado o mesmo mecanismo de GridFTP que é fornecido pelo GT2.

A Figura 6, ilustra melhor a diferença entre as terminologias usadas no desenvolvimento do GT2 e do GT3.

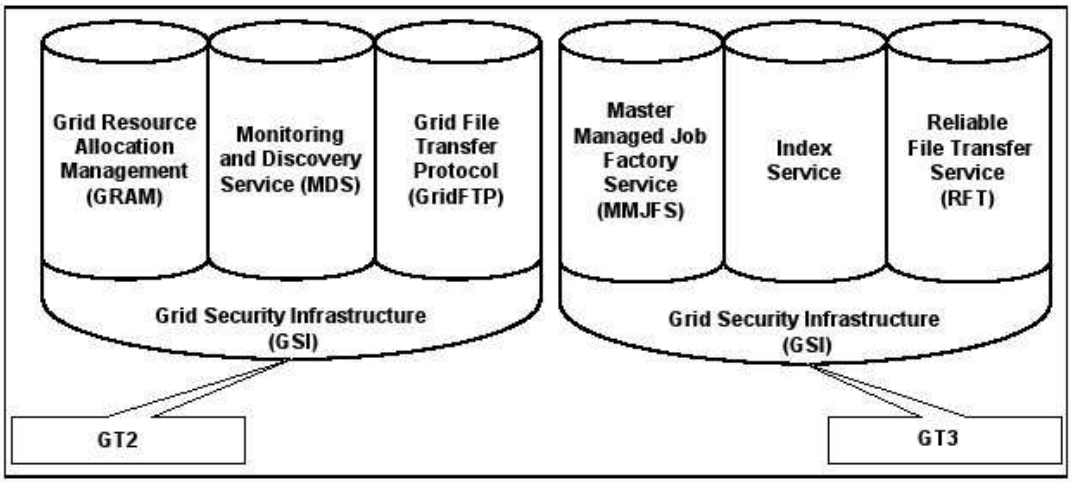

Figura 6 - Terminologia GT2 e GT3 (Ferreira 2004)

2.5.1.2 Commodity Grid Toolkits

Nos últimos anos diversas tecnologias tem surgido na área da computação distribuída com efeitos revolucionários na maneira como é acessada e processada a informação. Dentre estas tecnologias podem-se citar Web Services, Java e JINI[9], CORBA[10], DCOM[11], entre outras. Um desafio que os desenvolvedores tem enfrentado é unir estas tecnologias (chamadas doravante de "commodity") com as tecnologias emergentes dos Grids computacionais. Para auxiliar nesta tarefa, o projeto *Commodity Grid* foi desenvolvido para definir interfaces entre os Grids e frameworks já existentes. Usando-se uma definição mais formal, um Commodity Grid Toolkit (CoG Kit) define e implementa uma série de componentes que mapeiam as funcionalidades do Grid em um ambiente/framework commodity (Laszewski, 2004).

Desta forma, pode-se visualizar um CoG Kit para a tecnologia Web/CGI, um para Java, CORBA, DCOM, etc. Em cada caso o benefício do CoG Kit é facilitar ao desenvolvedor de aplicativos explorar serviços avançados do Grid (gerenciamento de recursos, segurança) enquanto utiliza componentes familiares providos pelos frameworks que já conhece.

Segundo Laszewski (2004), a palavra chave é *mapear*, pois a integração entre os

9 Arquitetura Java designada a prover serviços na rede e criar interações espontâneas entre aplicações usando estes serviços
10 Common Object Request Broker Architecture
11 Distributed Component Object Model

Grids e a tecnologia commodity não é simplesmente um problema de definição de interfaces, mas sim como os conceitos e serviços dos Grids podem ser melhor expressados em um framework particular. Por exemplo, no Globus Grid Toolkit o gerenciamento da computação remota é tratado usando-se uma API procedural enquanto que no Java CoG Kit a mesma funcionalidade é provida através de um objeto Job e eventos Java.

O projeto Commodity Grid tem desenvolvido CoGs para algumas linguagens conhecidas, dentre elas pode-se citar Perl e Python, para prover fácil prototipação e programação baseada em CGI para a Internet. O Java CoG foi desenvolvido para dar suporte a aplicações com interfaces gráficas e a habilidade de executar serviços Grids através de navegadores Web compatíveis. Além destas linguagens, devido a necessidade de acessar os serviços dos Grids usando-se frameworks de computação distribuída já consolidados no mercado iniciou-se o desenvolvimento de CoGs para CORBA e DCOM.

O exemplo abaixo demonstra como acessar serviços básicos de um Grid usando-se o Java CoG Kit (Laszewski, 2004). É um trecho de código de uma aplicação para processamento de dados climáticos.

```
// Passo 0. Inicialização
MDS mds=new MDS("www.globus.org",389,"o=Grid");

// Passo 1. Procura por uma máquina disponível
result = mds.search("(objectclass=GridComputeResource)(freenodes=64))","contact");

//Passo 1.a) Escolhe uma máquina
machineContact= <select the machine with minimal execution time from the contacts that are returned in result>

// Passo 2. Prepara os dados para o experimento
// Passo 2.a) Busca por dados climáticos e retorna
// os atributos: server,port,directory,file
dn = mds.search("(objectclass=ClimateData)(year=1999)(region=midwest","dn", MDS.SubtreeScope);
result = mds.lookup (dn,"server port directory file");

// Passo 2.b) Faz o download dos dados para a máquina
url = result.get("server" )+ ":"+ result.get(" port")+" :"+ result.get("directory")+"/"+ result.get("file");
data = server.fetch (url, machineContact);

// Passo 3. Prepara a descrição para executar o modelo
RSL rsl = new RSL("(executable=climateModel)(processors=64)(arguments=-grads)(arguments=-out map.grads)(arguments=-in" +
data.filename +")");

// Passo 4. Submete o programa
```

```
GramJob job = new GramJob();
job.addJobListener(new GramJobListener() {
   public void stateChanged(GramJob job){
      // react to job state changes
   }
}
try{
   job.request(machineContact, rsl);
}
catch (GramException e) {
   problem submitting the job
}
```

O trecho de código a seguir ilustra a utilização do pyGlobus, o CoG desenvolvido para a linguagem de programação Python (Jackson, 2004).

```
#Creating a job.
try:
   gramClient = GramClient.GramClient()
   callbackContact = gramClient.set_callback(func, condV)
   jobContact = gramClient.submit_request( clipper.lbl.gov ,  &(executable=/bin/sleep)(argument=15) , GramClient.JOB_STATE_ALL)
except GramClient.GramClientException, ex:
   print ex.msg
#Callback for state changes.
def func(cv, contact, state, error):
   if state == GramClient.JOB_STATE_PENDING:
      print "Job is pending"
   elif state == GramClient.JOB_STATE_ACTIVE:
      print "Job is active"
```

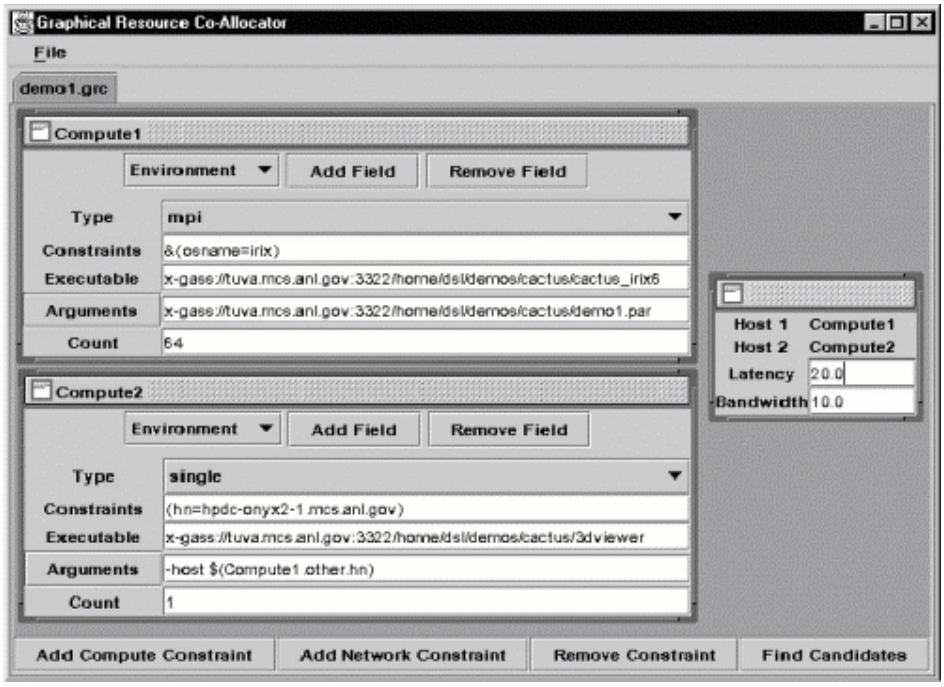

Figura 7 - Graphical Resource Co-Allocator (Laszewski 2004)

A Figura 7 mostra uma tela do Graphical Resource Co-Allocator, interface gráfica desenvolvida usando-se o Java CoG Kit. Esta ferramenta permite selecionar um recurso computacional para agendar uma tarefa em uma máquina que atenda algumas características definidas pelo usuário.

2.5.2 Legion

O ambiente de programação Legion é mantido pela *University of Virginia* e teve seu desenvolvimento iniciado em 1993. É um sistema baseado em objetos, onde tudo, desde a capacidade de processamento, memória e espaço de armazenamento é considerado como um objeto (Dantas, 2003). Sua principal idéia é uma arquitetura de grid provendo uma única máquina virtual para as aplicações do usuário, deixando algumas características do grid, como escalabilidade, tolerância à falhas e segurança totalmente transparentes para seus usuários finais (Dantas, 2003).

Também pode ser definido como uma camada intermediária que conecta redes, estações, supercomputadores e outros recursos juntos em um sistema abrangendo diferentes sistemas operacionais, arquiteturas e localizações físicas (Legion FAQ, 2004).

O Legion utiliza um serviço global de identificação, onde os objetos podem ser identificados por um mecanismo de nomeação em três níveis (Pitanga, 2003). 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 publica 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.

Os aspectos de segurança também são vistos pela perspectiva de objetos. O programador define a segurança associada a um objeto e quais tipos de mecanismos são permitidos neste, no próprio objeto, mais especificamente em seu LOID. Existe também um método chamado *May I*, que pode ser definido em cada classe de objetos para verificar o acesso permitido a estes. Quanto a segurança das mensagens trocadas entre os objetos, estas são criptografadas e assinadas usando métodos do sistema de chaves públicas RSA.

2.5.3 Globus x Legion

Tanto o Globus quanto o Legion são sistemas distribuídos de alta performance 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 (Legion FAQ, 2004).

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 grids, maior é a complexidade, principalmente utilizando-se a arquitetura do Globus.

3. Portais de Grid

Toda a tecnologia e teoria por trás dos Grids não teria sentido algum se a utilização por parte dos usuários não fosse a mais simples possível. Para o usuário a utilização de um grid de computadores para processar sua requisição deve ser transparente. Informações como quantas CPUs estão disponíveis, qual a velocidade e carga da rede devem ser disponibilizadas de uma forma que ele possa tomar certas decisões que melhorem a performance do processamento desejado.

Uma forma de se disponibilizar essas informações para os usuários do grid é através dos chamados Portais de Grids. Segundo Sun Microsystems (2004, p. 5), "um portal é um ambiente Web seguro que habilita uma organização a agregar e compartilhar conteúdo – informação, serviços e aplicações – com clientes, parceiros de negócios, empregados e fornecedores".

Grande parte dos usuários atualmente está familiarizada com a Internet e com todos os benefícios que a mesma fornece. Vários deles utilizam portais diariamente, sejam eles de conteúdo, como jornais e revistas, ou mais restritos, como portais empresariais ou Intranets. Baseando-se nesse conhecimento, os portais de grids fornecem uma interface amigável para seus usuários. Através desta interface é possível realizar operações como submeter um job ao grid, verificar o estado de um job em execução, ver um diagnóstico do grid, quais as CPUs ociosas, a velocidade e carga da rede, entre outras.

Outra vantagem que pode ser atribuída a um portal de grid é que o usuário pode submeter um job ao grid sem necessitar instalar nada em sua máquina, usando somente um navegador web. Toda a operação de criar um job e submetê-lo a uma CPU em específico- ou a mais de uma- fica a cargo do servidor web onde o portal está sendo executado.

Além disso, as tarefas de administração do grid também podem ser feitas remotamente, mediante uma validação no portal. A administração pode ser feita de uma maneira distribuída, delegando-se partes das tarefas à usuários distintos localizados em locais diferentes, tudo através de uma interface amigável, dispensando a execução de comandos ou a instalação de softwares específicos.

Como já citado, um mecanismo de validação é necessário para que os usuários tenham acesso as informações contidas nos portais. Além de restringir o acesso a informações e tarefas, mecanismos de contabilização também se fazem necessários para administrar o uso e a performance do portal e do grid como um todo. Protocolos criptografados como o

HTTPS[12] também são importantes para reforçar a segurança.

3.1 Ferramentas disponíveis

Diversas empresas fabricantes de software estão distribuindo ferramentas para o desenvolvimento de portais que se integrem com suas implementações de grids. Entre elas, pode-se citar a SUN, a Oracle e a IBM. Existem também algumas iniciativas abertas ou não vinculadas a uma empresa comercial unicamente. São projetos que tiveram seu início principalmente na área científica e acadêmica e que não estão vinculadas a estratégias comerciais de alguma grande empresa.

3.1.1 Grid Portal Development Kit (GPDK)

A idéia do GPDK é desenvolver componentes comuns que podem ser usados por desenvolvedores para construir um portal que pode autenticar seguramente usuários e ajudá-los a tomar decisões precisas no momento de agendarem jobs. Para isso, permite que os usuários tenham acesso a informações importantes sobre os recursos alocados no grid. Além disso, os usuários também podem visualizar e monitorar os jobs criados e seus resultados (GPDK, 2003).

12 HTTPS é a combinação dos protocolos HTTP com o SSL. Principal maneira de trafegar documentos via HTTP de maneira segura.

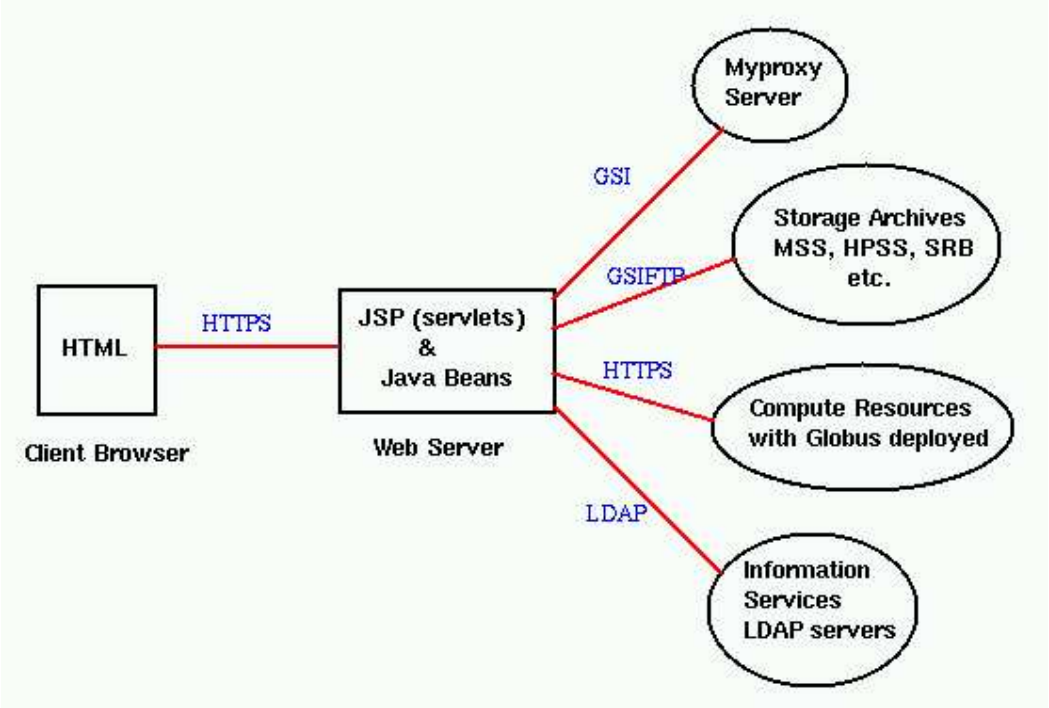

Figura 8 - Arquitetura do GPDK (GPDK 2003)

A Figura 8 demonstra a arquitetura do GPDK. Ela foi desenvolvida sobre o modelo de três camadas, onde um cliente utilizando um navegador web acessa, de modo seguro, via o protocolo HTTPS, 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/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 8, o GPDK tem seu funcionamento baseado em tecnologias Java, principalmente Java Server Pages (JSP) e Java Beans[13]. 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*

13 Java Beans é um componente Java reutilizável

(CoG) toolkit. O CoG provê uma série de APIs em Java para os recursos do Globus. 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.

Ainda sobre os beans do GPDK, segundo GPDK (2003), estes podem ser agrupados em cinco categorias, que são:

- Segurança: o principal bean é o MyproxyBean, que se comunica com o componente Myproxy. Este bean utiliza métodos para obter e definir o username, a senha e o tempo de vida desta credencial em um componente Myproxy;
- Perfis de usuários: Três beans são agrupados nesta categoria. O UserLoginBean, que provê um serviço opcional para validação de usuários no portal; UserProfileBean que guarda as informações de preferências, histórico de jobs e recursos computacionais de um usuário; e o UserAdminBean que armazena e valida um perfil de usuário especial, o perfil de administração;
- Submissão de jobs: o bean JobBean possui todas as informações necessárias no momento da submissão de um novo job, como requisitos de memória, nome e argumentos do executável, fila de execução, entre outros. Este bean é enviado a outro, conhecido como JobSubmissionBean que submete o job ao grid usando o *gatekeeper* do Globus. Outros beans importantes podem ser citados, como o JobInfoBean que contem as informações sobre o job submetido ao grid e o JobHistoryBean que armazena diversos JobInfoBean mantendo um histórico.
- Transferência de arquivos: a capacidade de transferência de arquivos é responsabilidade da interface FileTransferBean. Dois beans que implementam esta interface, o GSIFTPTranferBean e SISCPTransferBean realizam a cópia de arquivos de maneira segura entre as máquinas que fazem parte do grid.
- Serviços de informação: o bean MDSQueryBean contém os métodos necessários para realizar pesquisas em servidores LDAP em busca de informações como sistema operacional, carga de cpu e memória, entre outras.

3.1.2 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 padrões e softwares existentes para facilitar o acesso a infraestrutura do grid. O Legion Grid Portal pode ser definido como uma arquitetura para integrar um número de tecnologias existentes sobre uma interface comum (Natrajan, 2003).

Em sua versão atual utiliza o software Legion para interagir com os componentes do grid, mas é possível realizar algumas adaptações e utilizar o Globus ou outras ferramentas de grid. Na prática, quando um usuário utiliza a interface do portal para realizar alguma tarefa, o servidor executa as ferramentas de linha de comando do Legion. O portal suporta todos os comandos possíveis do Legion, em particular, a inicialização e monitoramento de aplicações e o acesso ao sistema de arquivos distribuído do Legion. Outros serviços como segurança, agendamento, transferência de dados, são suportados implicitamente, o que facilita o uso por parte dos usuários do portal.

A Figura 9 demonstra a arquitetura do Legion Grid Portal

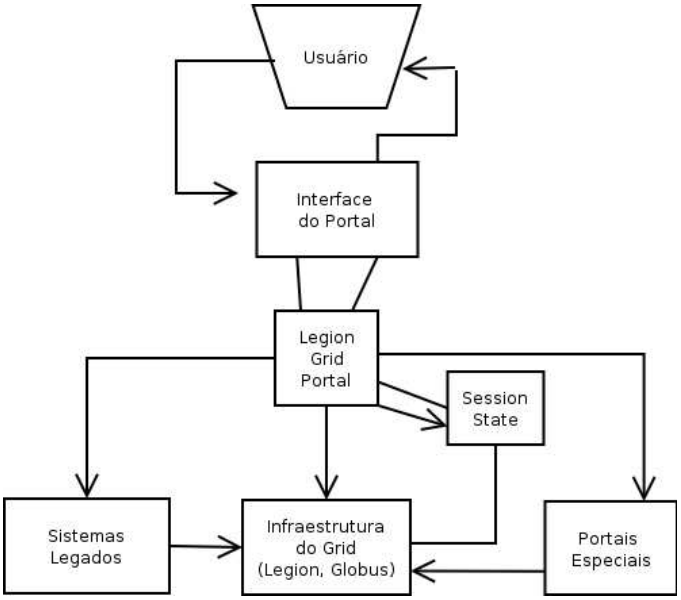

Figura 9 - Arquitetura do Legion Grid Portal (Natrajan 2003)

Como demonstra a figura, 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 CGI[14] desenvolvido em Perl, e tem como finalidade processar as requisições feitas pelos usuários através da interface.

14 CGI (*Common Gateway Interface*) é uma interface definida de maneira a possibilitar a execução de programas sob um servidor de informações. Pode ser desenvolvido em qualquer linguagem de programação, sendo que as mais comuns são C e Perl.

Durante a execução, o portal gera informações de sessão e cache, que são usadas para fins de autenticação e melhoria na performance das execuções. Essas informações, arquivos em cache como imagens e logs são armazenados no componente Session State. Nas versões atuais os Sistemas Legados citado na Figura 9, são bancos de dados e scripts necessários para o acesso a estes. Portais Especiais são usados para executar aplicações específicas através do portal, e armazenam ferramentas, software e scripts para estas aplicações. A Infraestrutura do Grid citada refere-se a serviços e ferramentas providas pelo Legion – ou o Globus- para gerenciar o grid.

Uma implementação do portal está em uso desde o ano 2000 em um grid chamado *npacinet*, que é o grid mundial do Legion.

3.1.3 Grid Portal Toolkit (GridPort)

O Grid Portal Toolkit (GridPort) é uma coleção de tecnologias utilizadas para auxiliar no desenvolvimento de portais científicos em grids computacionais, portais de usuários, interfaces de aplicativos e portais educacionais (Thomas, 2004).

As páginas e os dados são gerados por módulos e bibliotecas, desenvolvidas na linguagem de programação Perl, 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 utilizar para interagir com o Globus ou outras ferramentas de grid (Thomas, 2004).

Entre os usuários do GridPort pode-se citar a NASA, a *University of Southern Californa*, o *San Diego Supercomputer Center*, entre outros.

Na Figura 10 pode-se visualizar a arquitetura do GridPort.

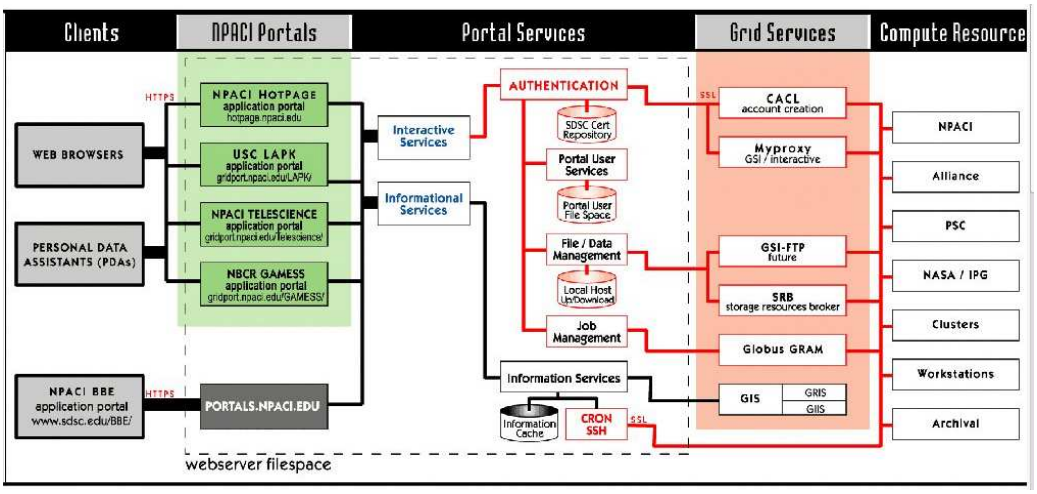

Figura 10 - Arquitetura do GridPort (Thomas 2004)

No diagrama da Figura 10, as diferentes partes do sistema são demonstradas na forma de camadas. 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 (Thomas, 2004).

Ainda sobre a Figura 10, as camadas podem ser descritas como:

- Clients: São navegadores web, PDAs ou outros dispositivos móveis. Nesta camada também figuram outros portais que podem se comunicar através de ferramentas como web services;
- NPACI Portals: são portais de aplicação que podem estar armazenados em outros servidores, mas que usam os mesmos componentes do GridPort. Isto permite que compartilhem dados, bibliotecas, arquivos;
- Portal Services: Além de realizar a mediação entre as requisições dos clientes e os serviços do grid, outros serviços são prestados para o portal e seus usuários, como gerenciamento de sessões, contabilização, coleções de arquivos e monitoramento. Além disso suporta a execução de comandos e a submissão de jobs através de módulos que podem ser facilmente expandidos para suprir a necessidade de portais específicos;
- Grid Services e Compute Resources: Nestas camadas estão os componentes responsáveis pela interação com as ferramentas do grid e os recursos alocados a este.

Os portais que utilizam a arquitetura acima descrita fornecem dois tipos básicos de serviços, que são os serviços informativos e os interativos. Os dados fornecidos pelos serviços informativos incluem estado das máquinas, carga e uso de filas, rede, etc. A maior parte

destas informações são coletadas através de scripts que executam em segundo plano ou utilizando a funcionalidade GIS fornecido pelo Globus Toolkit.

Os serviços interativos permitem que os usuários tenham acesso aos recursos computacionais do grid e permite que o servidor web realize as tarefas solicitadas pelos usuários nestes recursos.

3.1.4 Sun Technical Computing Portal

Como citado nos itens anteriores, grandes universidades e grupos de usuários estão desenvolvendo ferramentas próprias para criar seus ambientes de portais de grids computacionais. Além destas iniciativas, algumas empresas começam a visualizar oportunidades comerciais para esta tecnologia. Uma destas empresas é a Sun Microsystems, renomada na área de soluções de rede e desenvolvimento, que apresenta sua solução chamada Sun Technical Computing Portal (TCP).

Segundo Sun Microsystems (2004), o TCP é uma solução integrada para ambientes de computação em alta performance que provê uma nova classe de serviços web para as organizações. Baseada em outras ferramentas da Sun, como o Sun ONE Portal Server e no Sun Grid Engine, software gerenciador de recursos, a solução TCP fornece à organização uma interface simples e unificada para aplicações técnicas complexas. Tem por finalidade combinar as capacidades da tecnologia de portais, softwares de gerenciamento de recursos e ferramentas fáceis de utilizar para centralizar o gerenciamento de recursos computacionais e aplicações, assim como prover comunicação com alta segurança e facilitar o acesso por parte dos usuários.

A utilização de computação de alta performance pode ser caracterizada por aplicações não interativas para as quais o usuário carrega um arquivo de entrada e submete uma tarefa, como uma simulação ou consulta, e esta é executada por horas ou até dias. Na maneira tradicional, o controle e monitoramento destas tarefas requeriam o uso de mecanismos de segurança isolados para cada sistema além de conhecimentos específicos sobre o sistema operacional onde esta tarefa está executando. O TCP tem por finalidade eliminar esta complexidade, fornecendo aos usuários um único e seguro ponto de acesso, usando-se um navegador web, para todas as aplicações, tarefas e projetos. Através desta interface até mesmo usuários não técnicos podem submeter tarefas, monitorar sua situação e

acessar os resultados obtidos.

Dentre as principais vantagens que o TCP fornece, a SUN elenca como principais:

- Acesso facilitado: provê aos usuários acesso seguro em qualquer local. Tarefas podem ser submetidas local ou remotamente, e recursos computacionais estão disponíveis aos usuários de maneira simples;
- Melhor gerenciamento de recursos: o Sun Grid Engine computadores ociosos na rede e direciona seu poder computacional para atender as tarefas submetidas pelos usuários. Isto resulta em melhor aproveitamento do poder computacional da organização;
- Maior facilidade de uso dos recursos: torna a submissão e monitoramento tarefas fáceis. Com o TCP, os usuários podem gerenciar a submissão de tarefas preenchendo formulários baseados na Web, não necessitando conhecer detalhes complexos como scripts ou minúcias de sistemas operacionais. Aos usuários é apresentado formulários que lhe solicitam todas as informações necessárias, que são automaticamente convertidas em linhas de comando e executadas para que a tarefa tenha início. Mais tarde, os usuários podem utilizar o portal para verificar o status da execução da tarefa, receber notificações por e-mail quando a tarefa estiver completa, visualizar e copiar arquivos com resultados e compartilhá-los com outros;
- Fácil acesso a aplicações legadas: administradores podem facilmente adicionar aplicações legadas ao portal para compartilhamento e uso remoto. Em aplicações em lote que usam uma interface de linha de comando, geralmente nenhuma modificação é necessária para que se tornem disponíveis na Internet ou intranet, bastando simplesmente adicioná-las ao TCP;
- Soluções integráveis: habilita a organização a criar soluções baseadas em padrões e tecnologias abertas, como Java, HTML, XML e JavaScript, facilitando a interoperabilidade entre plataformas, sistemas e ambientes heterogêneos. Outra vantagem do TCP utilizar padrões abertos é que a organização ganha flexibilidade para adicionar aplicações e capacidades conforme a necessidade;

A solução TCP faz uso de uma arquitetura de três camadas, onde a camada *back-end* é o ambiente de computação de alta performance, como demonstra a Figura 11.

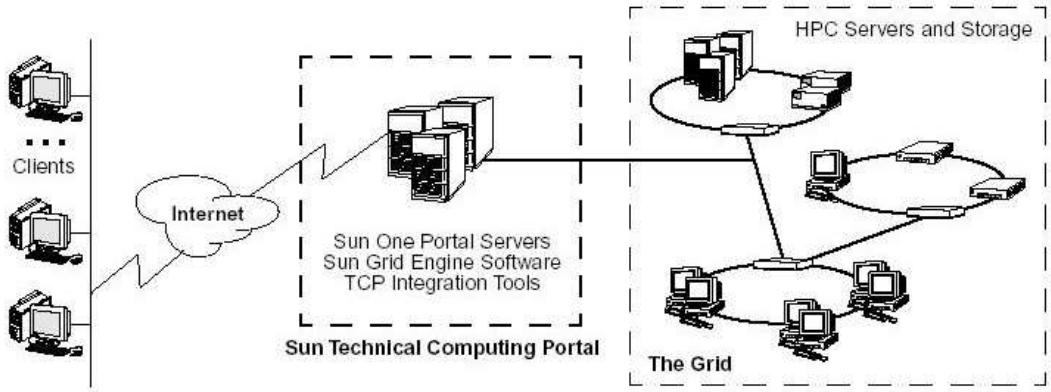

Figura 11 - Arquitetura do Sun TCP (SUN 2004)

A camada intermediária consiste de um ou mais Sun ONE Portal Server, cada um capaz de submeter tarefas ao Sun Grid Engine, que pode automaticamente encontrar o recurso computacional requerido na camada de *back-end*, ou seja, no Grid. E a primeira camada é constituída por um navegador web que toda a interface entre o usuário e o ambiente do Grid.

a. Sun ONE Portal Server

O Sun ONE Portal Server é uma solução de gerenciamento integrada para portais seguros. Usando capacidades de gerenciamento de identidades fornece mecanismos com os quais as organizações podem proteger suas informações enquanto constroem relações seguras entre comunidades de usuários como clientes, parceiros de negócios, fornecedores e empregados, usando-se uma interface Web familiar. Agrega conteúdo, aplicações e serviços que são personalizados baseados em papéis que os usuários desempenham, preferências dos usuários ou relevâncias determinadas pelo sistema.

Várias características contribuem para que o Sun ONE Portal Server seja uma solução para criar e manter portais produtivos e seguros:

- Gerenciamento integrado de serviços de portal e identidades, garantindo a capacidade de gerenciar e administrar usuários e políticas de acesso;
- *Single sign-on*, garante a flexibilidade da organização adicionar novos serviços sem aumentar a complexidade do acesso por parte dos usuários. Fornece acesso a toda a informação e aplicações agregadas sem a necessidade do usuário lembrar-se de senhas e logins individuais para cada recurso;

- Segurança avançada, provendo serviços de autenticação e autorização e reforçando regras e políticas de uso. Os métodos de autenticação podem variar de acordo com o grupo de usuários, mantendo o portal altamente seguro e flexível aos vários requisitos da organização;
- Busca e entrega segura, fornecendo documentos e recursos baseados nas políticas de acesso do usuário, garantindo que os arquivos serão entregues somente a quem possui permissão de acesso a eles;
- Capacidade de delegar funções administrativas a outros usuários, facilitando a administração;
- Ponto único de administração, provendo uma localização central para administrar e gerenciar o portal, buscas e identidades, diminuindo custos administrativos.

b. Sun Grid Engine Software

O Sun Grid Engine (SGE) é um software de gerenciamento de recursos distribuídos que auxilia na otimização da utilização de hardware e software em ambientes de rede heterogêneos. Agrega recursos computacionais e apresenta um único ponto de acesso aos usuários que necessitam destes recursos. Desta forma, o SGE pode aumentar a produtividade das máquinas e otimizar o número de tarefas que podem ser completadas.

O SGE aceita tarefas submetidas pelos usuários através da interface do Sun ONE Portal Server e agenda-os para a execução nos sistemas apropriados, de acordo com políticas de gerenciamento de recursos. Quando uma tarefa é aceita ela é enviada a um *scheduler*, que determina seus requisitos e prioridades e então é colocada em uma fila enquanto aguarda o início da execução. Um *daemon* inicia a execução das tarefas na fila, monitora seu progresso, reporta ao *scheduler* as condições de carga do servidor e avisa do final do processo.

O SGE provê as funções tradicionais de gerenciamento de recursos distribuídos e outras características específicas, como:

- Tarefas só são executadas quando os recursos que ela necessita estão disponíveis. Assim tarefas que fazem uso intensivo de certos recursos são processadas de maneira mais eficiente;
- Balanceamento de carga, distribuindo dinamicamente as tarefas para fazer melhor uso de todos os recursos computacionais disponíveis;

- Estatísticas de uso, fornecendo à organização condições de monitorar a utilização dos recursos e determinar como melhorar a alocação dos mesmos;
- Protocolo de encriptação baseado em certificados, melhorando a segurança usando a técnica de cifrar as mensagens usando-se chaves secretas;
- Gerenciamento de políticas, monitorando o progresso de todas as tarefas e ajustando suas prioridades de acordo com a política de administração.

c. Sun TCP Integration Tools

Usando-se o Sun ONE Portal Server é possível mostrar em uma página Web informações, aplicações e serviços. O Sun TCP Integration Tools fornece uma série de canais – que aparecem como menus ou janelas no navegador do usuário – que permitem ao usuário controlar e acessar as funcionalidades providas pelo SGE, integrando transparentemente suas capacidades de gerenciamento de recursos. Os seguintes canais são fornecidos por padrão pelo Sun TCP Integration Tools:

- *Project list*, provendo a capacidade dos usuários criarem projetos, onde podem carregar arquivos de entrada e gravar os arquivos de saída gerados pelas tarefas que forem submetidas e executadas;
- *Application list*, fornecendo aos usuários a lista das aplicações que foram disponibilizadas no portal pelo administrador ou outro usuário autorizado;
- *Job list*, onde os usuários podem submeter tarefas a aplicações e monitorar a execução das mesmas. Usuários podem também especificar um endereço de e-mail para serem notificados quando a tarefa tiver sua execução completa;
- *Application control*, fornece aos administradores e usuários autorizados a capacidade de adicionar aplicações ao portal e as disponibilizar para os usuários;
- *Job monitor*, fornece aos administradores opções de contabilização do uso do sistema.

A organização pode criar novos canais usando o *Channel Wizard*, assistente que faz parte do Sun ONE Portal Server. Usando-se o assistente para criar os canais algumas tarefas necessárias para a criação são automatizadas e facilitadas, como a nomeação do canal, a escolha das especificações e a adição do canal a lista disponível para os usuários.

3.1.5 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 grids computacionais baseado no conceito de *portlets*. Segundo o site GridSphere Portal Framework [15] "Gridsphere permite que desenvolvedores rapidamente desenvolvam e empacotem aplicações web portlets de terceiros de modo que possam ser executadas e administradas dentro do GridSphere portlet container".

Como citado, o principal conceito usado na criação do GridSphere é o de portlets. Segundo o GridSphere Portlet Reference Guide, portlets são definidos como "componentes visuais que podem ser assimilados dentro de páginas de um portal web. Eles provêem pequenos aplicativos que podem mostrar conteúdo informacional ou prover acesso a outros serviços". Com a utilização 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. Por exemplo, em um determinado portal podem existir diversos portlets, sendo um deles responsável por mostrar ao usuário notícias vindas de um site externo. O usuário pode "minimizar" o portlet como se fosse uma janela em um aplicativo de desktop, ocultando sua visualização. Ou, usando as opções de configuração do portlet, o usuário poderia alterar a fonte da notícia, buscando-as de outro site externo que desejar. Outra vantagem da utilização deste conceito é que novos portlets podem ser adicionados ao portal, fornecendo novas características e opções aos usuários.

A partir da versão 2.0 do GridSphere, dois modelos de desenvolvimento de portlets são suportados. A primeira, o modelo original 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 API[16], o que garante compatibilidade com aplicativos desenvolvidos por vários líderes de mercado, que também seguem esta especificação, tais como IBM, Sun, Oracle, entre outras.

A Figura 12 mostra um exemplo de portlet desenvolvido no GridSphere.

15 http://www.gridsphere.org

16 Java Specification Requests são padrões desenvolvido pela JCP (Java Community Process), que são grupos de usuários e empresas que definem os rumos da linguagem Java. O JSR 168 é a especificação que define as regras que permitem a interoperabilidade entre portlets e portais, definindo uma série de APIs relacionadas a agregação, personalização, apresentação e segurança. Mais informações sobre esta especificação podem ser encontradas no endereço http://www.jcp.org/en/jsr/detail?id=168

Figura 12 - Exemplo de GridSphere Portlet (Novotny 2004)

O GridSphere fornece alguns portlets em seu núcleo (core portlets) além de um ambiente de desenvolvimento que permite a criação de aplicativos seguindo os dois modelos acima citados.

Os componentes apresentados na Figura 13 são os componentes principais do

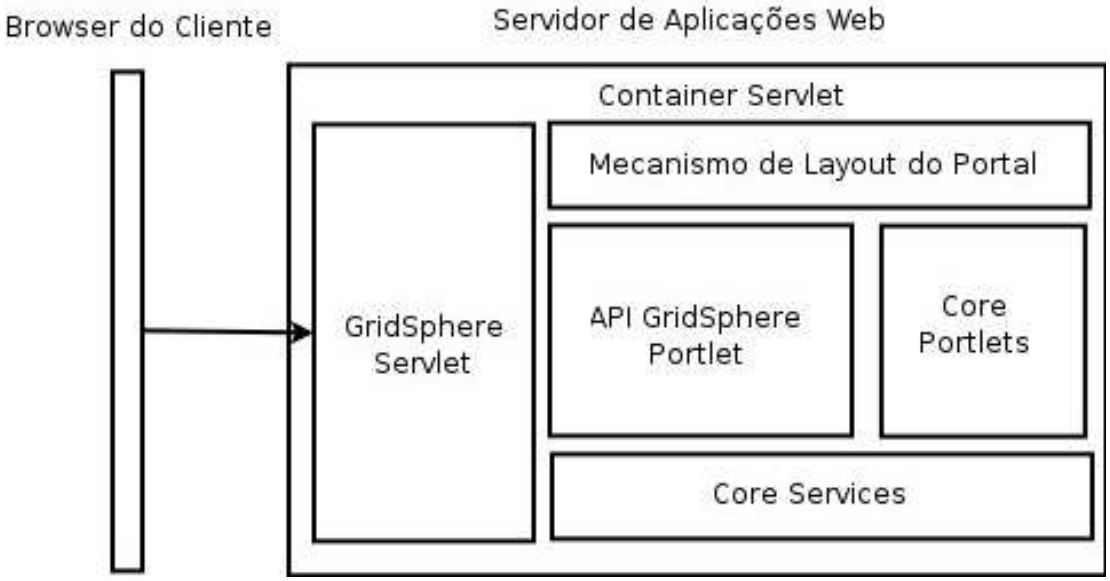

Figura 13 - Arquitetura do GridSphere (Novotny 2004)

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 renderize os dados formando a saída para o usuário.

Ambos, o servlet e o mecanismo responsável pela renderização do layout, fazem uso dos core portlets e services do GridSphere. Os principais portlets fornecidos para realizar requisitos básicos são, segundo Novotny (2004):

- *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:* fornce a interface com a qual um novo usuário pode requisitar a criação de uma conta no portal, podendo escolher também configurações avançadas como em qual grupo deseja se filiar.
- *Account Management:* permite que usários 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 interagir.
- *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 tem 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.

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 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). Um usuário com o papel de visitante pode visualizar apenas uma visão genérica e com poucas permissões do portal. No momento que este usuário realiza o login no portal, usando seu nome e senha, ele passa a executar o papel de um usuário, e o portal reflete as configurações contidas em seu perfil. Um usuário com o papel de administrador pode realizar configurações avançadas em todo o comportamento do portal, enquanto que o papel de super usuário lhe permite realizar qualquer ação dentro de todo o ambiente, sendo por isso sua utilização escolhida com extremo cuidado.

4. Considerações Finais

O presente estudo, demonstra as principais características e vantagens que a utilização dos Grids Computacionais podem trazer para as organizações. Nos últimos anos, a utilização dos grids tem ultrapassado as paredes dos laboratórios de pesquisas das universidades e avançado em direção ao mundo dos negócios. Mais e mais empresas tem visto grandes oportunidades que esta tecnologia tem a oferecer e apostam em seu crescimento. Empresas como IBM, Oracle, entre outras, tem investido milhares de dólares no desenvolvimento da tecnologia e em treinamentos para melhor aproveitar estas oportunidades. Este crescimento lembra muito o apresentado pela Internet em meados da década de 90, com a diferença que o mercado está muito mais maduro e a possibilidade de uma nova decepção como ocorrido na queda das "pontocom" parece cada vez mais distante.

Neste novo ambiente que a tecnologia está se desenvolvendo, a capacidade de crescimento e integração tem se mostrado extremamente importante. As organizações desejam aumentar suas capacidades tecnológicas, sejam elas de processamento, armazenamento ou comunicações, para melhor enfrentar o mercado concorrente, mas, ao mesmo tempo, necessitam integrar seus sistemas legados a esta nova estrutura e facilitar o acesso e treinamento de seus colaboradores e parceiros. Vindo ao encontro destas necessidades, a utilização de portais tem se mostrado muito eficaz.

Este estudo, realizou o levantamento de algumas das principais características dos portais de grids, além de detalhar algumas das ferramentas e implementações existentes hoje no mercado. A primeira conclusão a que se pode chegar ao fim deste trabalho é que as atuais ferramentas de desenvolvimento de portais, tanto as open source como as de código fechado, se encontram em um estado avançado, fornecendo boas opções aos desenvolvedores. Dentre as opções apresentadas, três chamam atenção. Duas opções de código aberto, contando com comunidades de desenvolvedores e baseadas em padrões bem definidos, a GridPort, implementada em grandes universidades como a Universidade do Texas, e a GridSphere. O Sun Technical Computing Portal é uma boa opção comercial, principalmente por contar com o suporte de uma grande empresa como a Sun.

Mas independente de solução, seja ela baseada em código aberto ou fechado, o mais relevante é a tecnologia envolvida nos portais de grids. Os antigos portais desenvolvidos pelas

equipes de cada projeto, com sua própria API e documentação tem sido substituídos por produtos baseados em padrões abertos e apoiados por grandes organizações. Um dos papeis fundamentais é desempenhado pelo conjunto de padrões que definem a criação e utilização dos Web Services. Na sua versão 3, o Globus Toolkit, que tem se mostrado a principal ferramenta para a construção da arquitetura básica de grids, é totalmente baseado no conceito de Grid Services, que por sua vez são pequenas modificações de Web Services. Seguindo esta tendência, as ferramentas de portais também passaram a se basear neste forte conceito, como o GridPort em suas versões mais novas e o GridSphere. Outra importante especificação, seguida pela ferramenta GridSphere que se mostra uma forte tendência é a utilização dos portlets. Graças a uma especificação da linguagem Java, já uma realidade e não tendência devido ao fato de que a maioria das ferramentas são baseadas nesta linguagem, é possível criar portais que podem interoperar e ser armazenados em servidores de aplicação de diversos fabricantes.

Como sugestão para trabalhos futuros pode-se citar estudos mais aprofundados nestas tecnologias que se mostram fortes tendências, como grid services e portlets. Além disto, pode-se estudar a criação de APIs que permitam a criação de portais de grids usando-se o advento dos Commodity Grids. Por exemplo, o desenvolvimento de uma API que permita a criação de portais usando-se o pyGlobus e armazenar estes no servidor de aplicações Zope. O servidor de aplicações Zope é baseado na linguagem de programação Python e é muito utilizado no desenvolvimento de aplicações web e sites de conteúdo, principalmente quando associado ao aplicativo Plone. Também é interessante o estudo da viabilidade da construção de um Commodity Grid para a linguagem de programação PHP, amplamente utilizada no desenvolvimento de aplicações web e portais.

Referências Bibliográficas

BERSTIS, V. **Fundamentals of Grid Computing**.
Disponível em: http://www.redbooks.ibm.com/redpapers/pdfs/redp3613.pdf.
Acesso em: Dezembro de 2003.

BOMBONATO, F. **Computação em Grid. Uma Introdução**.
Disponível em: http://www.geleira.org/pdf/grid_computing.pdf.
Acesso em: Dezembro de 2003.

CERQUEIRA, T. **Um estudo sobre portais para ambientes de grids computacionais.** 2003.

DANTAS, M, ALLEMAND, J. e PASSOS, L. **An Evaluation of globus and Legion Software Environments**.
Disponível em: http://hpcs2003.ccs.usherbrooke.ca/papers/Dantas_01.pdf.
Acesso em: Dezembro de 2003.

FERREIRA, L., BERTIS, V., ARMSTRONG, J., KENDZIERSKI, M., et al. **Introduction to Grid Computing with Globus**.
Disponível em: http://www.redbooks.ibm.com/redbooks/pdfs/sg246895.pdf
Acesso em: Dezembro de 2003.

FERREIRA, L., LUCCHESE, F., MINETTO E., YASUDA, T., et al. **Grid Computing on Research and Education.** Disponível em: http://www.ibm.com/redbooks. Acesso em: Dezembro de 2004.

FERREIRA,L. JACOB, B. SLEVIN, S, et al. **Globus Toolkit 3.0 Quick Start**. Disponivel em: http://www.redbooks.ibm.com/redpapers/pdfs/redp3697.pdf. Acesso em Dezembro de 2004.

FOSTER, I., KESSELMAN, C. e TUECKE, S. **The Anatomy of the Grid: Enabling Scalable Virtual Organizations.**
Disponível em: http://www.globus.org/research/papers/anatomy.pdf .
Acesso em: Novembro de 2003a.

FOSTER, I., KESSELMAN, C. **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: Novembro de 2003b.

FOSTER, I., KESSELMAN, C., TUECKE, S. e NICK, J. **The Physiology of the Grid: An Open Grid Services Architeture for Distributed Systems Integration.**
Disponível em: http://www.globus.org/research/papers/ogsa.pdf.
Acesso em: Novembro de 2003c.

GPDK. **Grid Portal Development Kit**.
Disponível em: http://doesciencegrid.org/projects/GPDK/.
Acesso em: Dezembro de 2003.

GridSphere Portlet Reference Guide.
Disponível em
http://www.gridsphere.org/gridsphere/docs/ReferenceGuide/ReferenceGuide.html.
Acesso em Dezembro de 2004.

JACKSON, Keith R. **pyGlobus: Python CoG Kit.** Disponível em http://www-unix.mcs.anl.gov/fl/events/agr2003/proceedings/D-jackson-pyGlobus.pdf. Acesso em Setembro de 2004.

LASZEWSKI, G., FOSTER, I., GAWOR, J. **GoG Kits: A Bridge between Commodity Distributed Computing and High-Performance Grids.** Disponível em http://www-unix.mcs.anl.gov/~laszewsk/papers/cog-final.pdf. Acesso em Setembro de 2004.

Legion FAQ.
Disponível em: http://legion.virginia.edu/FAQ.html.
Acesso em: Janeiro de 2004.

NASH, M. **Oracle 10g: Infrastructure for Grid Computing.**
Disponível em: http://otn.oracle.com/tech/grid/collateral/GridTechWhitePaper_final.pdf.
Acesso em Janeiro de 2004.

NATRAJAN, A., NGUYEN-TUONG, A., HUMPHREY, M., GRIMSHAW, A. **The Legion Portal**.
Disponível em: *http://legion.virginia.edu/papers/GCE01.pdf*.
Acesso em: Janeiro de 2003.

NOVOTNY, J., RUSSEL, M., WEHRENS, O. **GridSphere: An Advanced Portal Framework.**
Disponível em http://www.gridsphere.org/gridsphere/wp-4/Documents/France/gridsphere.pdf.
Acesso em Dezembro de 2004.

MCCOMMON, M. **Start to learn about Grid Computing.**
Disponível em: ftp://www6.software.ibm.com/software/developer/library/gr-starthere.pdf.
Acesso em: Janeiro de 2004.

MILEY, Michael. **The Grid. Bringing Computing Power to the Masses.** Stokie: Oracle Magazine, número 5, Vol. XVII, pag. 39-44, 2003.

PITANGA, Marcos. **Computação em Cluster – O Estado da Arte em Computação**.Rio de Janeiro: Editora Brasport, 2003.

SUN MICROSYSTEMS. ***Sun Technical Computing Portal.*** Disponível em http://sun.com/solutions/hpc/pdfs/TCP-final.pdf . Acesso em Abril de 2004.

THOMAS, M., DAHAN, M., MUELLER, K., MOCK, S., MILLS, C., REGNO, R. **Application Portals: Practice and Experience**.
Disponível em: http://tacc.utexas.edu/~mthomas/pubs/GridPort_Apps_CPE.pdf.
Acesso em: Janeiro de 2004.

TUECKE, S., FOSTER, I., KESSELMAN, C., et al. **Open Grid Services Infrastructure (OGSI)**. Dispoível em : http://www.ggf.org/ogsi-wg. Acesso em Dezembro de 2004.