Computação em Grade e o Globus Toolkit 4

Professores:

Patricia Kayser Vargas[6]
(kayser@unilasalle.edu.br)
Marluce Rodrigues Pereira[7]
(marluce@dcc.ufla.br)
Claudio F. Resin Geyer[8]
(geyer@inf.ufrgs.br)

Resumo:

Os ambientes computacionais em grade (*computational grid*) são uma importante alternativa para a execução de aplicações de alto desempenho. Através da interligação de redes locais e *clusters* dispersos geograficamente, é possível construir um ambiente com um grande número de recursos heterogêneos que são acessados de forma transparente. O Globus Toolkit é um dos ambientes pioneiros, sendo constantemente referenciado na literatura especializada. Um dos objetivos do Globus é se firmar como um padrão. O foco deste tutorial é apresentar a importância da computação em grade bem como a utilidade do Globus mostrando os principais detalhes da versão 4.

6 Graduada em Ciência da Computação pela Universidade Federal do Rio Grande do Sul. Mestre em Ciência da Computação pela Universidade Federal do Rio Grande do Sul. Doutora em Engenharia de Sistemas e Computação pela Universidade Federal do Rio de Janeiro. Atualmente é professora do Centro Universitário La Salle. Atua principalmente nas áreas de sistemas distribuído, grid computing e escalonamento distribuído.

7 Graduada em Informática pela Universidade Federal de Juiz de Fora. Mestre em Engenharia de Sistemas e Computação pela Universidade Federal do Rio de Janeiro. Doutora em Engenharia de Sistemas e Computação pela Universidade Federal do Rio de Janeiro. Atua principalmente nas áreas de programação paralela e distribuída, processamento paralelo, programação lógica com restrições e processo de desenvolvimento de software

8 Doutor em Informática pela Université Joseph Fourier, Grenoble, 1991. Professor do Instituto de Informática da UFRGS desde 1981. Professor orientador de mestrado e doutorado do PPGC. Área de Pesquisa: Processamento Paralelo e Distribuído. Temas de interesse: ambientes de programação distribuída-paralela, em especial com orientação a objetos; sistemas de controle de redes de agregados (GRID); sistemas distribuídos adaptativos; escalonamento e computação móvel.

4.1 Introdução

O termo *grid computing* ou computação em grade surgiu em meados da década de 1990, como sendo uma infra-estrutura de computação distribuída para executar aplicações científicas e de engenharia que demandavam alto poder computacional [FOS98, FOS01].

Um ambiente de computação em grade permite compartilhar e coordenar o uso de recursos distribuídos geograficamente. Estes recursos podem ser: CPUs, sistemas de armazenamento, conexões de rede ou qualquer outro recurso de software ou hardware disponível.

Um dos primeiros projetos que motivaram a utilização de computação em grade foram as pesquisas realizadas em *High Energy Physics* (HEP), que requerem o processamento de grandes quantidades de dados usando algoritmos de processamento intensivo.

Outra área de pesquisa que gera muita informação a ser processada é a bioinformática, em aplicações como seqüenciamento genômico de vários organismos.

Analisando essas duas áreas, pode-se inferir que as principais motivações para o surgimento da computação em grade são:

- Existência de muitos dados para serem processados (*e.g.* grandes bases contendo dados coletados de um acelerador de partículas);
- Limitação de recursos locais (*e.g.* um determinado departamento pode ter máquinas insuficientes para comparar dados em tempo hábil para decidir novos experimentos);
- Possibilidade de negociar a utilização de recursos em conjunto com outras instituições (*e.g.* uma determinada instituição pode ceder ciclos de CPU em troca de espaço de armazenamento, ou de ciclos de CPU no futuro).

Existem vários trabalhos que buscam implementar um *middleware* para a construção de ambientes de grade como, por exemplo, Globus [GLO06], Condor [CON06], Legion [GRI03], GrADS [GRA06], OurGrid [OUR06], EasyGrid [EAS06] e GRAND [GRA06].

Nestes e em outros trabalhos, existem problemas que ocorrem em qualquer sistema distribuído, mas que são mais complexos em ambiente de grade devido a heterogeneidade e escala. Por exemplo, gerenciamento de recursos [NAB03, KRA02], gerenciamento de dados [STO01, BOS05], questões econômicas (*grid economy*) [BUY05], gerenciamento da rede [SAN03], segurança[WEL03] e gerenciamento de aplicação [DUT03, SAN05, VAR05a].

Neste tutorial, trataremos das soluções propostas pelo Globus. O Projeto Globus [GLO06] é um dos projetos de tecnologia para grade mais referenciados na literatura. O Globus Alliance é a organização que conduz as pesquisas e o desenvolvimento, tendo como principais participantes o Argonne National Laboratory, na University of Southern California's Information Sciences Institute, e na University of Chicago. Além disso, envolve outras instituições de pesquisa como a University of Edinburgh, o Swedish Center for Parallel Computers e o National Center for Supercomputing Applications (NCSA) e empresas com a IBM, Microsoft e Cisco.

Um dos trabalhos desenvolvidos no contexto deste projeto é o Globus Toolkit [FOS98a, GLO06]. O Globus Toolkit (GT) é um software livre que fornece diversos serviços que podem ser usados para a construção de sistemas de grades computacionais. O GT é um *toolkit* que consiste de um conjunto de componentes que implementam serviços básicos e de alto nível, tais como segurança, alocação de recursos, gerenciamento de dados e comunicação.

Diversos projetos de pesquisa utilizam alguma versão do GT como parte da sua infra-estrutura de grade. Alguns exemplos são: Enabling Grids for E-Science in Europe (EGEE) [EGE06], TeraGrid [TER06], Access Grid [ACC06], Grid Physics Network (GriPhyN) [GRI06], International Virtual Data Grid Laboratory [IVD06], eDiaMoND [EDI06], Particle Physics Data Grid [PPD06], e European Data Grid [DAT06].

O objetivo deste texto é apresentar uma visão geral sobre o Globus e os principais aspectos relacionados ao Globus Toolkit 4 (GT4), servindo como uma introdução ao tema. A idéia é que este material sirva de referencial teórico e conceitual para o tutorial a ser ministrado no ERAD2007. Aspectos mais "práticos" ou "operacionais", embora possam ser tratados durante a apresentação, não foram incluídos neste material. Assim, inicialmente apresenta-se os principais conceitos relacionados à computação em grade (Seção 4.2) e uma perspectiva histórica das versões do Globus Toolkit (Seção 4.3). Depois, trata-se de alguns detalhes do GT4 e os principais componentes são apresentados (Seção 4.4). O texto é finalizado com algumas considerações finais (Seção 4.5).

4.2 Computação em grade

Para tornar este mini-curso auto contido, isto é, sem exigir como pré-requisito o conhecimento prévio sobre computação em grade, esta seção apresenta uma visão geral. O objetivo é familiarizar com os conceitos e nomenclatura, permitindo o entendimento das seções seguintes. Leitores familiarizados com o assunto podem avançar diretamente para a Seção 4.3.

Em primeiro lugar, é preciso apresentar os principais termos relacionados diretamente a grade. Pode-se definir uma **grade** (*grid*) como "uma coleção de recursos computacionais heterogêneos, geograficamente espalhados, e pertencentes a diferentes domínios administrativos" [VAR05].

Uma infra-estrutura de grade deve prover os recursos heterogêneos e geograficamente distribuídos requisitados pelas aplicações. Assim, um **ambiente de grade** é "uma infraestrutura que deve fornecer facilidades para que aplicações de diferentes usuários possam acessar estes recursos da forma mais transparente possível" [VAR05].

Segundo Baker *et al.* existem três principais aspectos que caracterizam uma grade computacional [BAK00]:

- *heterogeneidade*: um grade envolve uma multiplicidade de recursos que são heterogêneos por natureza e que podem estar dispersos por numerosos domínios administrativos através de grandes distâncias geográficas;

- *escalabilidade*: uma grade pode crescer de poucos recursos para milhões. Isto levanta o problema da potencial degradação do desempenho à medida que o tamanho de uma grade cresce. Conseqüentemente, aplicações que requerem um grande

número de recursos dispersos geograficamente devem ser projetadas para serem extremamente tolerantes à latência;

- *dinamicidade ou adaptabilidade*: em uma grade, a falha de um recurso é a regra, e não a exceção. De fato, com tantos recursos, a probabilidade de que algum recurso falhe é naturalmente alta. Os gerenciadores de recursos ou aplicações devem adaptar o seu comportamento dinamicamente a fim de extrair o máximo de desempenho a partir dos recursos e serviços disponíveis.

A pesquisa de computação em grade está fortemente relacionada às áreas de ambientes distribuídos e redes. Um ambiente de grade difere de um sistema distribuído convencional pelo seu foco em compartilhamento em grande escala de recursos, aplicações inovadoras e, em alguns casos, busca por alto desempenho e/ou alta vazão (*high-throughput*) [FOS01, LAF02].

Assim, para compreender a utilidade e a importância do GT, deve-se ter presentes pelo menos dois aspectos sobre grades: (1) um ambiente de grade envolve recursos heterogêneos e de diferentes organizações; e (2) a coordenação desses recursos não é trivial, envolvendo uma série de tarefas. Por isso, construir um *middleware* de grade é uma tarefa árdua que envolve a criação de vários serviços básicos (e.g. autenticação de usuário) e de mais alto nível (e.g. um portal para submissão de tarefas). O GT tenta auxiliar nesta tarefa fornecendo uma série de componentes e serviços que podem ser combinados para a criação de uma infra-estrutura completa.

Aplicações

A característica principal para que uma aplicação seja caracterizada como uma aplicação que necessita de um ambiente de grade para ser executada é que o tamanho e/ou a complexidade do problema requer que pessoas em diversas organizações colaborem e compartilhem recursos computacionais, dados e instrumentos. Desta forma, as aplicações podem ser classificadas em três classes: de computação intensiva, uso intensivo de dados e de colaboração distribuída.

As aplicações de **computação intensiva** referem-se a problemas de simulação interativa como modelagem climática; simulação e análise de larga-escala, como formação da galáxia, detecção de onda de gravidade e simulação de eventos em campo de batalha; e engenharia, como estudos de parâmetros, modelos de componentes ligados.

Aplicações que fazem **uso intensivo de dados** referem-se a problemas de análise de dados experimentais (física de altas energias); e análise de imagem e sensor, usados em astronomia, estudos de clima e ecologia.

Os problemas de **colaboração distribuída** estão ligados a instrumentação on-line (microscópios e dispositivos de raio-X); visualização remota (em estudos climáticos e de biologia); e engenharia (teste estrutural de larga escala e engenharia química).

Também podemos dizer que essas aplicações são "grandes desafios" (*grand challenge applications*). Este tipo de aplicação trata de problemas fundamentais em ciências e engenharia com um amplo impacto econômico e científico. Sua principal característica comum é a alta demanda por recursos computacionais tais como ciclos de CPU e/ou armazenamento de dados [ANL06]. Assim, também é interessante caracterizar as aplicações

de grade pela área de conhecimento a que estão associadas. Duas áreas cons-tantemente referenciadas são bioinformática e física de altas energias (HEP), que possuem vários projetos que vêm construindo e utilizando a infra-estrutura de grade para realização de suas pesquisas.

A busca por similaridade entre seqüências de proteínas e ácidos nucléicos é uma operação importante e comum em **bioinformática**. Os banco de dados de seqüências têm crescido imensamente e continuam a crescer de forma rápida. Devido à magnitude dos problemas, a estratégia de comparação de seqüências deve ser otimizada. FASTA [FAS06] é uma técnica de alinhamento de seqüência que usa heurísticas para prover buscas mais rápidas do que estratégias mais exatas, que são baseadas em técnicas de programação dinâmica. Dado o tamanho dos bancos de dados, é indesejável transportar ou replicar todos os bancos de dados de todos os nodos em uma grade distribuída. FASTA possui uma implementação que usa banco de dados distribuídos e remotos que são parcialmente replicados em alguns nós da grid. FASTA é estruturado no modelo mestre-escravo e implementado em C [FAS06]. Desta forma, o gerenciamento de dados no ambiente de grade mostra-se fundamental para este tipo de aplicação.

Existem também projetos para HEP onde estimasse-se que serão processados de 12 a 14 Petabytes de dados por ano. Um dos projetos [BUN03] tem como objetivo encontrar mecanismos responsáveis pela massa no universo e partículas associadas à geração de massa. Estão envolvidos cerca de 2000 cientistas em 150 instituições situadas em mais de 30 países, caracterizando a existência de um volume muito grande de informações, sobre um mesmo tema de pesquisa, que estão distribuídos em todo o mundo.

Além disso, existem algumas aplicações distribuídas referenciadas na literatura que podem ser utilizadas para avaliação de grades:

- ScaLAPACK é um pacote de software popular para álgebra linear paralela, incluindo a solução de sistemas lineares baseados em LU e fatorações QR.

- JACOBI usa iteração Jacobi para resolver um sistema de equações lineares. Uma porção de vetor desconhecida x é atribuída a cada processador. Durante cada iteração, todo processador computa novos resultados para sua porção de x e então realiza *broadcast* de sua porção atualizada de x para todos os outros processadores. Jacobi é uma aplicação que utiliza memória principal de forma intensa com uma fase de comunicação envolvendo mensagens pequenas. Desta forma, um gerenciamento de recursos de forma adequada se faz necessário.

- Fish é uma aplicação que modela o comportamento e interações de peixes e é indicativo de muitas aplicações de físicas de partículas. A aplicação calcula as forças de Van der Waals entre partículas num campo bi-dimensional. Cada processo é responsável por um número de partículas que movem sobre o campo. A quantidade de computação depende da localização e proximidade das partículas. Fish exibe a quantidade dinâmica de trabalho por processador.

- Game of Life implementa Conway's Game of Life; um bem conhecido autômato celular binário. Uma malha bi-dimensional de pixels é usada para representar um ambiente de células. Em cada iteração, toda célula é atualizada com um stencil de 9-pontos e então os processadores enviam dados de suas arestas para seus vizinhos na

malha. Game of Life requer uma quantidade de memória significativa comparada com suas necessidades de computação e comunicação.

- Simulações de Monte Carlo [GRI87], que são parte de muitas aplicações de HEP, tem esse nome devido ao uso de números aleatórios nos seus algoritmos. Os chamados experimentos de Monte Carlo são amostras de experimentos, obtidas computacionalmente, usualmente realizadas para determinar a distribuição estatística esperada considerando um conjunto de pressupostos (*assumptions*) probabilísticos.

4.3 Evolução do Globus Toolkit

Um dos objetivos desta seção é apresentar uma perspectiva histórica da evolução do GT. Isso permitirá que o leitor compreenda as diferenças entre as versões e saiba selecionar o melhor material de acordo com a versão que estiver interessado em trabalhar. É importante notar que, apesar de todo o esforço da Globus Alliance de estimular a migração dos usuários para a nova versão, existe projetos que continuam trabalhando com versões anterior do GT.

Conforme visto na Seção 4.2, pode-se entender uma grade computacional como um conjunto de recursos computacionais interligados por uma rede e gerenciados por um *middleware*. O Globus toolkit (GT) é uma das ferramentas para a construção deste *middleware*, buscando tornar o acesso aos recursos o mais transparente possível. O projeto do GT seguiu os seguintes princípios [GLO06]:

- baixo custo;
- controle local, isto é, permitir controle dentro dos domínios;
- suporte para adaptação;
- modelagem seguindo o modelo de ampulheta (vide Figura 4.1) em analogia ao adotado protocolo IP (*"IP hourglass" model*).

Roure *et al* [ROU03] identifica três gerações na evolução dos sistemas de computação em grade:

- *primeira geração* na qual estão os precursores da computação em grade, tal como reconhece-se atualmente, sendo composto por projetos para integração de centros de super-computação (*supercomputing sites*). Tais propostas eram denominadas metacomputação e datam do início à metade da década de 90.
- *segunda geração* com o foco em *middleware* para suporte à dados e computação em larga escala. Caracterizada principalmente por projetos que tratam de infraestrutura, serviços chaves, aplicações específicas e portais de domínio;
- *terceira geração* é a geração corrente onde a ênfase muda para questões de colaboração global distribuída, abordagem orientada a serviços e camadas de informação.

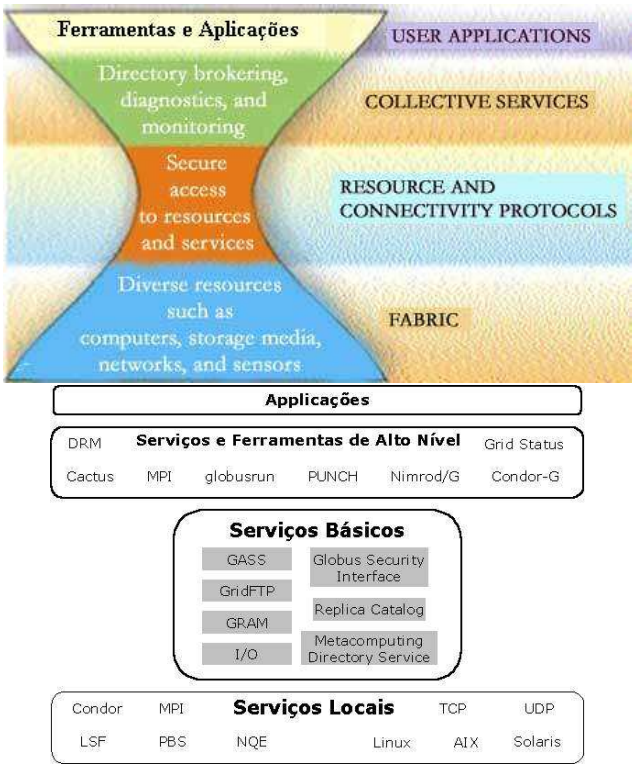

Figura 4.1: Globus Toolkit: modelagem seguindo o modelo da ampulheta.

Neste relativo curto período de existência, o GT passou por 4 versões e dezenas de releases. Nota-se, em cada uma das versões, mudanças significativas na organização e forma de disponibilização dos serviços, o que nos permitiria classificar as versões do Globus em função das gerações propostas por Roure *et al.*.

Um dos projetos representativos da primeira geração é o I-WAY (*The Information Wide Area Year*) [DEF96]. Ele foi concebido no início de 1995 como uma rede experimental de alto desempenho interligando vários computadores de alto desempenho e ambientes avançados de visualização. Dentre os problemas tratados neste projeto, cita-se questões de comunicação, gerenciamento de recursos e manipulação de dados remotos, para serem capazes de trabalhar eficientemente e efetivamente. O I-WAY é considerado como um precursor de projetos como o Globus e o Legion [GRI99, GRI03].

O Projeto Globus lançou a primeira versão do Globus Toolkit (GT1) em 1998. No entanto, os pesquisadores da Globus Alliance consideram o ano de 1996 como o marco de início do GT, uma vez que neste período foi concedido o primeiro apoio financeiro (através de um projeto patrocinado pelo Departamento de Defesa) e começaram a ser feitas as primeiras implementações. Deste modo, o ano de 2006 marcou a primeira década do projeto Globus [YUR06].

A versão 2.0 (GT2), de 2002, possui seu projeto fortemente relacionada à arquitetura proposta por Foster *et al.* [FOS01]. Tanto GT1 quanto GT2 podem ser considerados como de segunda geração, tanto pelo período de desenvolvimento quanto pela forma de modelagem do software.

Finalmente, como de terceira geração podemos classificar as duas versões mais re-

centes do Globus. O Globus Toolkit versão 3 (GT3) é baseado em *grid services*, que são muito parecidos com *Web services*. O GT3 implementa o *Open Grid Service Infrastructure* (OGSI) [FOS02]. Em 2004 foi lançada a especificação da *WS-Resource Framework* (WSRF) que propôs mudanças na OGSI, usando conceitos de *Web services*. O WSRF foi usado na construção da versão 4 (GT4) cujo lançamento da primeira versão estável ocorreu em 2005.

GT2 GT1 foi utilizado por pioneiros na pesquisa em grade, mas com o lançamento do GT2 como um software mais maduro e estável, ainda que com problemas, estimulou a utilização do Globus em diversos projetos. Ainda hoje alguns projetos ainda utilizam o GT2, ou componentes que existiam no GT2 dentro de versões posteriores.

O GT2 se propõe a auxiliar a implementação de um *middleware* de grade composto em camadas conforme proposto por Foster *et al.* [FOS01] Considerando essa arquitetura de referência proposta por Foster *et al.*, teríamos os seguintes componentes em cada camada do sistema:

- *infra-estrutura* (*fabric*): composto por computadores individuais, Condor pools, sistemas de arquivos, redes, sensores, etc. Poucas são as restrições impostas à tecnologia de baixo nível, sendo definida apenas interfaces e não características físicas.

- *conectividade*: com relação a comunicação, propõe o uso de protocolos padrões da Internet (IP, DNS, routing, etc.). Com relação a segurança, GT2 propõe uma primeira versão de *Grid Security Infrastructure* (**GSI**) que realiza autenticação, autorização e mecanismos de proteção de mensagens de forma uniforme. O GSI permite *single sign-on* e delegação, utilizando tecnologia de chave pública, SSL, X.509 e GSS-API.

- *recurso* (*resource*): esta camada, no contexto do GT2, contém os seguintes serviços: *Grid Resource Allocation Management* (**GRAM**), que permite alocação remota, reserva, monitoração e controle dos recursos computacionais; protocolo **GridFTP** que permite acesso de alto desempenho aos dados; e *Grid Resource Information Service* (**GRIS**) que disponibiliza acesso a informações sobre infra-estrutura e estado do sistema. Nesta camada também existem mecanismos para reserva, monitoração e controle da rede. Todos estes elementos são construídos utilizando a camada de conectividade, mais especificamente o GSI e IP.

- *coletiva* (*collective*): nesta camada existiriam serviços como *checkpoint* distribuído, gerenciamento de tarefas e replicação que não estão contemplados na distribuição do GT2.

No entanto, é importante ressaltar que esta visão modular é característica do GT2, embora vários desses componentes estejam disponíveis nas versões posteriores. A Figura 4.2 mostra os principais componentes que fazem parte do GT2. Nota-se que os principais componentes que compõem o GT2 são o GSI (na figura como "Pre-WS Authentication Authorization"), o GridFTP, o GRAM, o MDS e biblioteca em C. Esses componentes são denominados como "Non-WS Components" porque se referem justamente a componentes construídos sem seguir o padrão de *Web services* (WS).

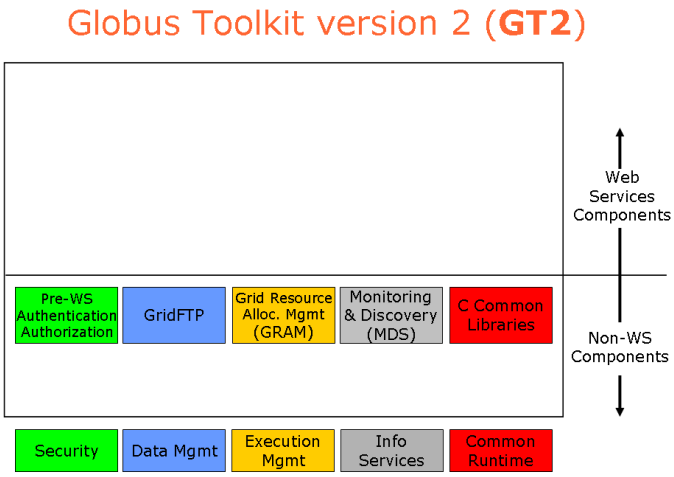

Figura 4.2: GT2: principais componentes [GLO06].

Note também que nesta figura, assim como nas demais figuras que mostram os componentes do GT, existe uma divisão por categorias. Logo abaixo do quadrado que engloba os componentes, existem cinco quadrados que servem como uma legenda para indicar a categoria ao qual cada coluna de componentes pode ser considerada. Maiores explicações sobre cada uma destas categorias são apresentadas na Seção 4.4.

GT3 Seguindo as tendências atuais de boa construção de software, o GT a partir da versão 3 usa a noção de arquitetura orientada a serviços [PAP03]. Aplicações orientadas a serviço pode disponibilizar a aplicação como um serviço, bem como organizar o grafo da aplicação na forma de um *workflow*.

Como o GT3 [FOS02a] apresenta uma abordagem de serviços, os vários componentes do *toolkit* são vistos como serviços que podem ser disponibilizados aos clientes, não seria usual encaixar os serviços na estrutura de camadas de Foster *et al.* [FOS01]. A invocação de componentes de outras camadas é análoga a construção de um serviço pela composição de vários serviços. No entanto, a principal diferença é conceitual: por isso a partir da GT3, a classificação dos componentes e serviços passou a ser feita por categoria e não mais por camadas.

O núcleo da infra-estrutura do Globus Toolkit 3 (GT3) é construído em cima das primitivas e protocolos da *Open Grid Service Infrastructure* (OGSI) [FOS02] que é uma implementação *open source* do *Open Grid Service Architecture* (OGSA) [TUE02].

O núcleo do GT3 oferece um ambiente de execução capaz de hospedar serviços de grade (*Grid services*). Um serviço de grade (*Grid service*) geralmente significa qualquer serviço oferecido a clientes em um ambiente de grade. No contexto do GT3, um serviço de grade tem que ser necessariamente compatível com a especificação da OGSI e deve apresentar a sua interface através da linguagem de descrição de serviços Web (*Web Services Description Language* ou WSDL).

O OGSI foi projetado para permitir a composição de serviços baseado em um conjunto de primitivas de computação distribuída. Ou seja, ao invés de especificar um

conjunto de serviços de alto nível completo, um conjunto de serviços primitivos são especificados e podem ser usados para construir e compor novos serviços.

A arquitetura do núcleo do GT3 pode ser visualizada na Figura 4.3. A Figura 4.4 é simular a Figura 4.2, mas mostrar quais componentes do GT2 foram mantidos, bem como quais os novos componentes incluídos. Nota-se que alguns serviços incluído são WS, enquanto outros não.

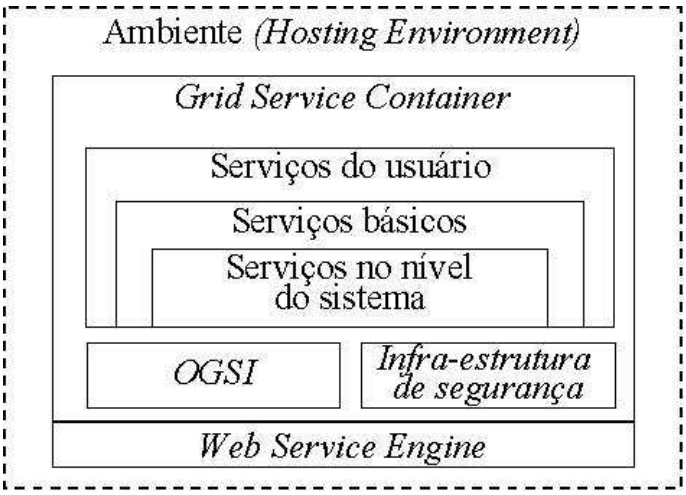

Figura 4.3: Arquitetura do GT3

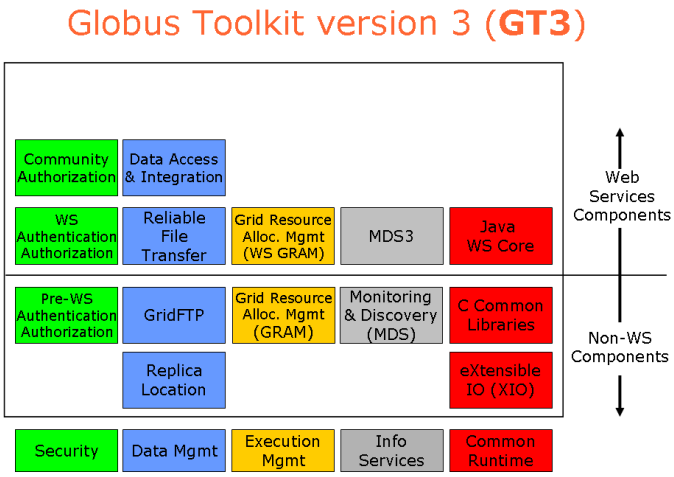

Figura 4.4: GT3: principais componentes [GLO06].

A implementação da Infra-estrutura de segurança (*Security Infrastructure*) prove proteção de mensagens no nível de transporte, autenticação mútua ponto a ponto e serviço de autorização *single sign-on*. Ela é basicamente uma adaptação do GSI da GT2 para um ambiente OGSI. A infra-estrutura de segurança e a implementação de referência OGSI

são dois blocos básicos que não fornecem nenhum serviço de execução mas servem de base para os outros serviços.

Os Serviços no nível do sistema são serviços no nível de execução que são genéricos o suficiente para serem usados por e em conjunto com outros serviços. O GT3 também fornece uma série de serviços de mais alto nível, isto é, Serviços básicos como por exemplo serviço de informação.

Todos estes serviços e primitivas, juntamente com os Serviços Definidos pelo Usuário, interagem com um ambiente de execução OGSI chamado *Grid Service Container*. O objetivo deste container é encapsular a aplicação evitando que ela tenha que tratar de configurações de execução específicas. Também serve para controlar o ciclo de vida dos serviços e o disparo de requisições remotas.

Finalmente, o *Web Service Engine* e o container estão hospedados em um Ambiente (*hosting environment*) que implementa as funcionalidades de um servidor Web tradicional.

GT4 e WSRF O grupo de pesquisa Globus e a IBM propuseram o Web Service Resource Framework (WSRF) [WS-04]. Eles lançaram uma primeira arquitetura e especificação em janeiro de 2004, em co-autoria com HP, SAP, Akamai, TIBCO e Sonic. O WSRF é uma extensão da OGSI composto com um conjunto de seis especificações de *Web services*. A partir dessa especificação, o GT4 foi concebido e implementado. Esta versão será detalhada na próxima seção.

4.4 Globus Toolkit 4

O Globus Toolkit pode ser visto como uma coleção de serviços e protocolos que pode ser utilizado de forma integrada ou em combinação com outras soluções. De fato, não parece haver uma versão do Globus Toolkit que funcione completamente sem a utilização de software adicionais, pelo menos para fazer o escalonamento dos recursos.

Web Services, *Grid Services* e WSRF

Pode-se caracterizar um **serviço** como uma funcionalidade de um sistema que é disponibilizada para os clientes em uma forma padronizada (como um serviço). Um *Web service* (WS) é um tipo particular de serviço. ***Web services*** ou serviços Web podem ser definidos como "aplicações baseadas na Web que utilizam padrões abertos e baseados em XML e protocolos de transporte abertos para a troca de dados com os clientes que os invocam" [MIC06].

Segundo Sotomayor e Childers [SOT05], um *Web service* não guarda estado, ou seja, é *stateless*, pois isso o torna mais flexível e mais de acordo com as recomendações da W3C. No entanto, para muitas aplicações de grade, é preciso que o serviço guarde seu estado (seja *statefull*). A proposta da WSRF é disponibilizar a partir de um *stateless Web service* um "*statefull Web service*". Para fazer isso sem precisar fazer mudanças significativas na infra-estrutura de *Web services*, o WSRF utiliza a abstração de recurso para armazenar o estado. Assim, pode-se definir um **Grid service** como um *Web service* associado a um recurso segundo algumas normas padronizadas. Note que neste contexto,

um **recurso** é uma entidade que guarda informações de estado de um determinado recurso computacional, possuindo um identificador único.

Também é importante deixar claro o que significa o padrão WSRF (*Web Service Resource Framework*) da OASIS. Ele é um conjunto de especificações que permite especificar *statefull Web services*, extendendo a noção de *Web services*. Segundo Sotomayor e Childers [SOT05], enquanto o OGSA é a arquitetura utilizada pelo Globus, a WSRF é a infra-estrutura na qual esta arquitetura é construída.

Principais Componentes

O GT4 possui uma série de componentes fracamente acoplados que são ilustrados nas Figuras 4.5 e 4.6. A Figura 4.5 mostra os principais componentes da versão 4, de forma simplificada e sem indicar a existência de componentes de versões anteriores ("non-WS"). Pode-se dizer que esses componentes compõem cinco categorias ou áreas de domínio [SOT05]:

- *Segurança*: responsável pela aplicação uniforme de políticas através de domínios distintos. O conjunto de componentes responsáveis por aspectos de segurança é conhecido como *Grid Security Infrastructure* (GSI). Este grupo é composto por quatro componentes: *Authentication Authorization* (controle de acesso a serviço e recursos e uso de diferentes métodos de autorização), *Community Authorization* (políticas de autorização para recursos de organizações virtuais (VOs)), *Delegation* (permite delegação de credenciais) e *Credential Mng* (inclui o SimpleCA e MyProxy que serão discutidos posteriormente).

- Gerência de Dados: trata da descoberta, transferência e acesso a dados. Principais componentes são *Grid FTP* (protocolo otimizado para a transferência de grandes quantidade de dados entre nodos), *Reliable File Transfer* ou RFT (usa o Grid FTP para prover transferência confiável), *Data Access & Integration* ou OGSA-DAI (usado para integrar dados armazenados em formatos distintos), *Replica Location* ou RLS(permite que o usuário saiba da localização de réplicas de *datasets* existente em diferentes nodos) e *Data Replication* (usa RLS e RFT para disponibilizar cópias de *datasets*).

- Gerência de execução: trata do *deployment*, escalonamento e monitoramento de tarefas. Os principais componentes são: *Grid Resource Allocation & Management* ou GRAM (principal componente, responsável pelo *deployment* e monitoramento), *Workspace Management* (permite criar e gerenciar dinamicamente áreas de trabalho em recursos remotos), *Community Scheduling Framework* ou CSF (provê uma interface única para diferentes escalonadores de recurso) e *Grid Telecontrol Protocol* (interface de serviço para controle remoto de instrumentos).

- Serviços de Informação: permite descoberta e monitoramento de serviços dinâmicos. O conjunto destes componentes é referenciado usualmente como MDS (*Monitoring and Discovery System*). *Index* (coleta dados dos recursos, sendo utilizado para agregar recursos de interesse), *Trigger* (coleta dados dos recursos e pode

realizar determinadas ações em função dos dados) e *WebMDS* (permite visualizar os dados coletados através de um navegador).

- Núcleo de execução (*Common Runtime*): Infra-estrutura para construção de novos serviços, existindo principalmente três tipos *Java Runtime*, *C Runtime* e *Python Runtime*. Esses componentes incluem ferramentas e bibliotecas de desenvolvimento bem como um ambiente para hospedagem de serviços.

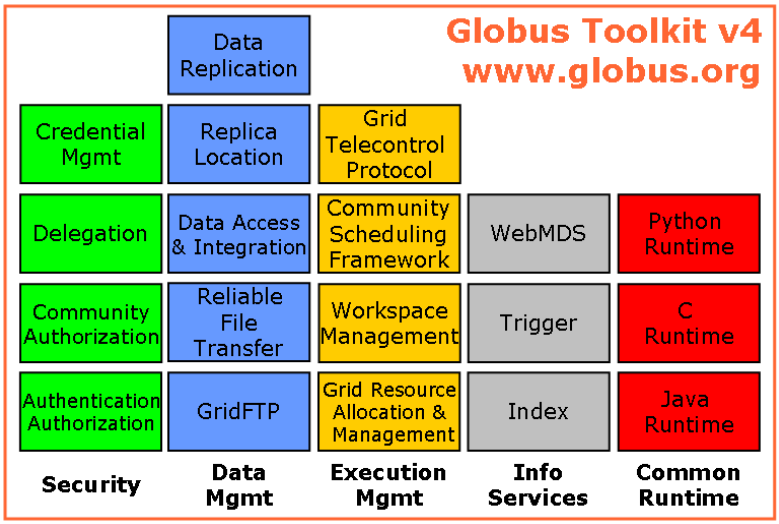

Figura 4.5: GT4: principais componentes [GLO06].

A Figura 4.6 é uma outra versão sobre os componentes, em que é apresentado um detalhamento: os componentes do GT2 e GT3 que foram mantidos, bem como os novos componentes incluídos. Os serviços são divididos em componentes *Web services* e em não *Web services* (também referenciados na literatura com pré-WS).

Alguns desses componentes serão apresentados nas próximas seções. Mas antes do detalhamento de qualquer um desses componentes, o texto apresenta uma pequena discussão sobre importantes conceitos relacionados com GT4, em especial o conceito de *Grid service*.

Gerência de Execução: GRAM Uma das parte centrais de um sistema distribuído é o gerenciador de recursos (RMS ou *Resource Management System*). O gerenciamento de recursos visa gerenciar um conjunto de recursos que estão disponíveis e que podem incluir recursos de diferentes provedores. Os recursos gerenciáveis em uma grade são principalmente processadores, banda de rede e espaço de armazenamento.

O ambiente de grade introduz vários problemas ao gerenciamento de recursos [CZA98]. Um dos problemas encontrados em gerenciamento de recursos é a autonomia nos sítios, pois os recursos podem pertencer a organizações diferentes (domínios diferentes). Além disso, os sítios podem usar diferentes gerenciadores de recursos locais. Outro problema, é que um gerenciador de recursos deve suportar o desenvolvimento de novos mecanismos de gerenciamento específicos para um determinado domínio

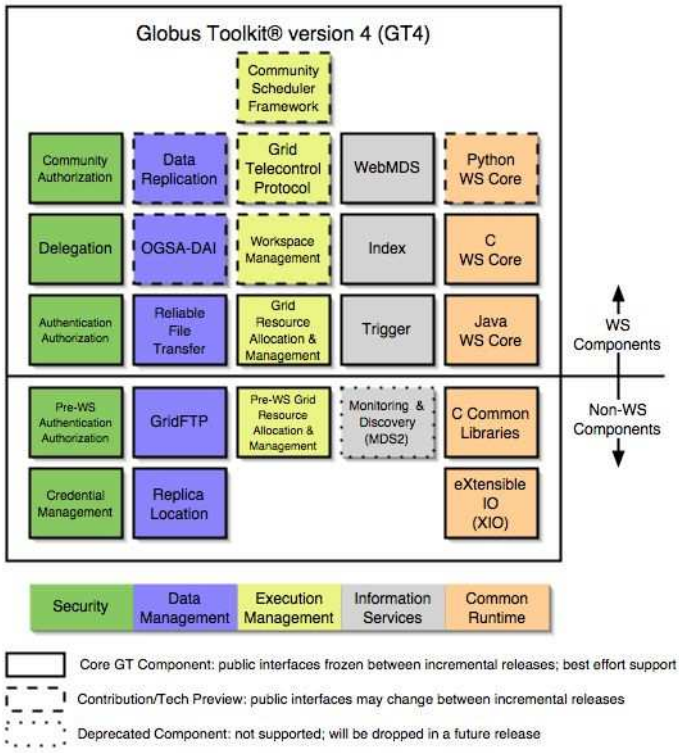

Figura 4.6: GT4: componentes WS e não-WS [GLO06a].

de aplicação sem exigir mudanças no código instalado nos sítios participantes. Deve, também, suportar negociação para adaptar requisitos da aplicação à disponibilidade de recursos. Desde as primeiras versões do Globus, o GRAM esteve presente [GLO06b], mas evolui bastante ao longo do tempo.

O WS GRAM [GLO06c] é um software que implementa uma solução para o gerenciamento de tarefas dentro do contexto do GT4. Ele apresenta interfaces de WS consistente com o modelo WSRF.

A idéia do GRAM é suportar a execução de tarefas usando serviços de escalonamento e adaptadores para escalonadores locais, ao mesmo tempo que faz o envio dos dados de entrada de forma coordenada (*file staging*).

Em princípio o GRAM funciona apenas para sistemas operacionais que sigam o modelo de programação e segurança do UNIX.

O escalonamento pode ocorrer através de um escalonador de tarefas locais em um determinado nodo da grade. Mas o GRAM também possui a habilidade de criar processos usando o método padrão *fork()* do UNIX. De qualquer modo, o mais provável é que conjuntos grandes de recursos estejam sobre o controle de escalonadores como por exemplo PBS, LSF, Loadleveler ou Condor.

Outro aspecto característico de padrão UNIX utilizado é o utilitário *sudo*. Utilizando este comando, o WS GRAM ganha privilégios, sem precisar se tornar super usuário, através do qual é possível realizar operações específicas do sistema operacional para inicializar e executar as tarefas do usuário.

Para realizar suas tarefas, o GRAM precisa interagir com outros serviços do GT4

[GLO06b]:

- ReliableFileTransfer (RFT): invocado para realizar (*file staging*) antes e após a execução da tarefa. Essa é uma das principais contribuições da versão GT4 do GRAM: ao utilizar o RFT, agrega-se ao serviço de gerência uma maior robustez no envio de dados e um melhor suporte para recuperação em caso de falhas.

- GridFTP: de fato, o RFT utiliza o GridFTP para prover o seu serviço de transferência confiável, logo, é natural que ambos estejam relacionados ao GRAM. Um ponto importante é que, embora o GridFTP não precise estar rodando no mesmo servidor que encontra-se o GRAM, o GRAM de alguma forma precisa ter acesso ao GridFTP. Isso se deve ao fato de que o (*file staging*) somente é possível para arquivos que encontram-se compartilhados com um servidor GridFTP que esteja registrado no serviço GRAM.

- Delegation: usado por clientes para delegar credenciais para uso dos serviços GRAM ou RFT. A idéia do uso da Delegação permite uma maior flexibilidade, permitindo, por exemplo, que uma credencial dê acesso apenas a tarefas de uma determinada aplicação.

Gerência de dados: GridFTP O GT4 apresenta diversos serviços e componentes para gerência de dados. A seguir, o GridFTP será apresentado por ser um elemento que vem sendo proposto e aprimorado ao longo dos anos, e por ser utilizado como base para diversos outros serviços do GT4.

GridFTP [ALL02, MAN05, GLO06d] é um protocolo otimizado para redes de longa distância usado para transferência com alto desempenho, segurança e confiabilidade de dados. O protocolo GridFTP é baseado no FTP que é um protocolo de transferência de arquivos na internet altamente popular. Algumas funcionalidades foram selecionadas de características e extensões já definidas no padrão FTP bem como novas características foram adicionadas para atender aos requisitos dos atuais projetos de *data grid*.

Um funcionalidade importante provida pelo GridFTP é a "transferência com três participantes" (*third party transfer*): um cliente organiza a movimentação de dados entre dois servidores GridFTP remotos.

GridFTP foi definido como um padrão através de recomendações do Global Grid Forum (GGF). Ele provê transferência segura, robusta e eficiente de dados. No GT4, existem implementações deste protocolo, disponibilizando principalmente [GLO06d]: (1) uma implementação de servidor, chamado *globus-gridftp-server*; (2) um cliente por linha de comando, chamado *globus-url-copy*; e (3) um conjunto de bibliotecas de desenvolvimento.

Allcock *et al.* [ALL05] apresenta alguns detalhes da versão do GridFTP para o GT4, composta principalmente por um *Globus striped GridFTP framework* e um servidor. Este *framework* implementa o protocolo, inclusive com relação ao suporte para transferências distribuídas independentes de múltiplas fontes de dados (*stripped transfer*) e detecção de falhas. Ambos os componentes fazem parte do GT4 e servem de base para outros serviços relacionados a segurança e gerência de dados. Esta versão do GridFTP

é baseada no sistema *Globus eXtensible Input/Output* (XIO) [GLO06e] e possui suporte para IPv6.

Alguns itens que foram considerados no projeto do GT4 GridFTP [ALL05]:

- *stripping*: suporte a operações de movimentação *stripped* de dados, isto é, na qual dados distribuídos através de um conjunto de repositórios remotos é transferido para um outro conjunto de sistemas de armazenamento;

- operações coletivas: ao tratar dos envios independentes como sendo uma operação coletiva, o sistema busca oportunidades para otimizações no envio dos dados;

- interfaces uniformes: qualquer tipo de repositório deve poder ser acessado de forma equivalente;

- rede: utilizar protocolos padrões de comunicação, em especial TCP ou variações seguras ou eficiente de TCP.

É importante observar que, apesar das várias melhorias no GT4 GridFTP, ele ainda possui compatibilidade de protocolo e API com relação a GT3.2 [GLO06d].

Segurança: MyProxy Desde as primeiras versões do GT, os mais diversos aspectos de segurança são tratados, buscando com isso facilitar a construção de *middlewares* de grade mais seguros, confiáveis e robustos. O conjunto de componentes e serviços que tratam diretamente de segurança recebem o nome de GSI (*Grid Security Infrastructure*).

Alguns conceitos chaves de segurança tratados pelo GT4, segundo [SOT05], são os seguintes:

- comunicação segura: mais do que apenas enviar dados criptografos, é preciso garantir três aspectos para obter uma comunicação segura: privacidade, integridade e autenticação;

- privacidade: apenas o emissor e o receptor de uma mensagem são capazes de entender a comunicação; normalmente obtido através de algoritmos de criptografia e decriptografia;

- integridade: o receptor tem certeza de que a mensagem recebida é exatamente igual a enviada pelo remetente; isso pode ser obtido através do uso de esquemas de chave pública (*public key encription*);

- autenticação: é preciso garantir que as partes envolvidas na comunicação sejam quem elas estão se identificando, protegendo o sistema contra usuários maliciosos; algoritmos modernos de criptografia são capazes de garantir a autenticação;

- autorização: é um mecanismo que decide quando um usuário está autorizado a realizar uma determinada tarefa.

Para esses conceitos, no GT4, a base reside na utilização de certificados (*X.509 end entity certificates*): (1) identificam entidades persistentes como usuários e serviços; (2) proporcionam para cada entidade um identificador único; (3) garantem a identificação de um usuário ou serviço (através dos princípios de criptografia de chave pública empregados pelo certificado); (4) dão segurança adicional ao estabelecer que cada certificado possui um prazo de validade.

No esquema de chaves e certificados do GT, seria necessário, em princípio, copiar os certificados e chaves privadas de cada usuário para cada máquina pertencente a grade. Um dos serviços disponíveis no GT4 é o MyProxy[GLO06f]. A grosso modo, podemos definí-lo como um repositório de credenciais (*X.509 proxy credential*) que elimina a necessidade de copiar a chave privada e os arquivos de certificados manualmente para os recursos.

4.5 Conclusões

O presente texto apresentou uma introdução aos principais conceitos de computação em grade e uma visão geral do Globus Toolkit (GT). Atualmente, a computação em grade apresenta-se como uma das infra-estruturas mais significativas e promissoras no contexto de computação de alto desempenho. Conforme mencionado, este tipo de infra-estrutura é a base de desenvolvimento de aplicativos para inúmeros projetos nas mais diversas áreas de conhecimento.

Desde a sua concepção inicial, o GT não teve por objetivo ser um *middleware* completo. Como o próprio nome diz ele é um *toolkit* formado por diversos componentes relativamente autônomos. Existem vários exemplos de projetos que utilizam o GT como parte da sua infra-estrutura de grade.

Acredita-se que novas aplicações com Globus sejam cada vez mais desenvolvidas com uso do GT4 devido as diversas vantagens que a versão atual possui. No entanto, é interessante saber as principais características destas versões anteriores, uma vez que ainda existe projetos utilizando versões do GT2 ou GT3. O principal motivo para que esses usuário não migrem imediatamente para o GT4 é a mesma de outros contextos onde exista um custo associado à migração de software: até que ponto vale a pena portar um código legado estável para a nova versão do software. Além disso, o estilo de programação do GT2 para o GT4 mudou significativamente e, em alguns casos, há falta de programadores familiarizados com os conceitos de *Web* e/ou *Grid services*.

Recomendamos aos leitores interessados que procurem as referências bibliográficas constantes no final deste documento para uma visão mais aprofundada do tema. Em especial, sobre questões operacionais e programação em Java com o GT4 recomendamos o livro de Sotomayor e Childers [SOT05].

Os autores deste mini-curso gostariam de agradecer a diversos colegas que contribuíram, de forma direta ou indireta, na confecção deste material. Em especial, aos pesquisadores da UFRGS, Luciano, Éder e Diego, e da UFRJ, Inês, Bernardo e Vinícius.

Referências

[ACC06] ACCESS grid project. [S.l.: s.n.], 2006. (`http://www.accessgrid.org`).

[ALL02] ALLCOCK, W.; BRESNAHAN, J.; FOSTER, Ian; LIMING, L.; LINK, J.; PLASZCZAC, P. GridFTP protocol specification. [S.l.: s.n.], 2002. (Technical Report. Available at `http://www.globus.org/research/papers/GridftpSpec02.doc`).

[ALL05] ALLCOCK, William; BRESNAHAN, John; KETTIMUTHU, Rajkumar; LINK, Michael. The globus striped gridftp framework and server. In: , 2005, Washington, DC, USA. **Anais...** IEEE Computer Society, 2005. p.54.

[BAK00] BAKER, Mark; BUYYA, Rajkumar; LAFORENZA, Domenico. The Grid: international efforts in global computing. In: , 2000, Rome, Italy. **Anais...** 2000.

[BOS05] BOSE, Rajendra; FREW, James. Lineage retrieval for scientific data processing: a survey. **ACM Computing Surveys**, New York, NY, USA, v.37, n.1, p.1–28, 2005.

[BUN03] BUNN, Julian J.; NEWMAN, Harvey B. Data intensive grids for high energy physics. In: BERMAN, Francine; FOX, Geoffrey; HEY, Tony (Eds.). **Grid computing**: making the global infrastructure a reality. [S.l.]: Wiley & Sons, 2003. p.859–906.

[BUY05] BUYYA, Rajkumar; ABRAMSON, David; VENUGOPAL, Srikumar. The grid economy. **Proceedings of the IEEE**, v.93, n.3, p.698–714, March 2005. (Special Issue on Grid Computing).

[CON06] CONDOR homepage. [S.l.: s.n.], 2006. (`http://www.cs.wisc.edu/condor/`).

[CZA98] CZAJKOWSKI, Karl; FOSTER, Ian; KARONIS, Nicholas; KESSELMAN, Carl; MARTIN, Stuart; SMITH, Warren; TUECKE, Steve. A resource management architecture for metacomputing systems. In: , 1998. **Anais...** 1998. p.62–82.

[DEF96] DEFANTI, Thomas A.; FOSTER, Ian; PAPKA, Michael E.; STEVENS, Rick; KUHFUSS, Tim. Overview of the I-WAY: wide-area visual supercomputing. **The International Journal of Supercomputer Applications and High Performance Computing**, v.10, n.2/3, p.123–131, Summer/Fall 1996.

[DUT03] DUTRA, Inês de Castro; PAGE, David; SANTOS COSTA, Vitor; SHAVLIK, Jude; WADDELL, Michael. Toward automatic management of embarrassingly parallel applications. In: , 2003, Klagenfurt, Austria. **Anais...** 2003. p.509–516.

[EDI06] EDIAMOND – diagnostic mammography national database project. [S.l.: s.n.], 2006. (`http://www.ediamond.ox.ac.uk`).

[EGE06] EGEE – enabling grids for e-science in europe. [S.l.: s.n.], 2006. (`http://public.eu-egee.org;`).

[FAS06] FASTA sequence comparison at the university of virginia. [S.l.: s.n.], 2006. (`http://fasta.bioch.virginia.edu/`).

[FOS02] FOSTER, Ian; KESSELMAN, Carl; NICK, Jeffrey; TUECKE, Steven. The physiology of the Grid: an open grid services architecture for distributed systems integration. [S.l.: s.n.], 2002. (Disponível em `http://www.globus.org/research/papers/ogsa.pdf`).

[FOS02a] FOSTER, Ian; KESSELMAN, Carl; NICK, Jeffrey; TUECKE, Steven. Grid services for distributed system integration. **IEEE Computer**, v.35, n.6, June 2002.

[FOS01] FOSTER, Ian; KESSELMAN, Carl; TUECKE, Steven. The anatomy of the Grid: enabling scalable virtual organizations. **The International Journal of High Performance Computing Applications**, v.15, n.3, p.200–222, Fall 2001.

[FOS98] FOSTER, Ian; KESSELMAN, Carl. **The Grid**: blueprint for a new computing infrastructure. San Francisco, California, USA: Morgan Kaufmann, 1998. p.701.

[FOS98a] FOSTER, Ian; KESSELMAN, Carl. The Globus project: a status report. In: , 1998. **Anais...** 1998. p.4–18.

[GLO06] GLOBUS alliance. [S.l.: s.n.], 2006. (`http://www.globus.org/`).

[GLO06a] GLOBUS. About the globus toolkit. [S.l.: s.n.], 2006. (`http://www.globus.org/toolkit/about.html`).

[GLO06b] GLOBUS. Gt execution management: gram. [S.l.: s.n.], 2006. (`http://www.globus.org/toolkit/gram/`).

[GLO06c] GLOBUS. Gt 4.0 ws gram approach. [S.l.: s.n.], 2006. (`http://www-unix.globus.org/toolkit/docs/4.0/execution/key/WS_GRAM_Approach.html`).

[GLO06d] GLOBUS. Gt 4.0 GridFTP. [S.l.: s.n.], 2006. (`http://www.globus.org/toolkit/docs/4.0/data/gridftp/`).

[GLO06e] GLOBUS. Gt 4.0 XIO. [S.l.: s.n.], 2006. (`http://www.globus.org/toolkit/docs/4.0/common/xio/`).

[GLO06f] GLOBUS. Gt 4.0: credential management: MyProxy. [S.l.: s.n.], 2006. (`http://www.globus.org/toolkit/docs/4.0/security/myproxy/`).

[GRA06] GRAND - grid robust application deployment. [S.l.: s.n.], 2006. (`http://www.cos.ufrj.br/grand/`).

[ANL06] GRAND challenge applications. [S.l.: s.n.], 2006. (`http://www-new.mcs.anl.gov/grand-challenges/`).

[GRI87] GRIER, David Alan. Systems for monte carlo work. In: , 1987. **Anais...** ACM Press, 1987. p.428–433.

[GRI99] GRIMSHAW, A.; FERRARI, A.; KNABE, F.; HUMPHREY, M. Legion: an operating system for wide-area computing. **IEEE Computer**, v.32, n.(5), 1999.

[GRI03] GRIMSHAW, Andrew S.; NATRAJAN, Anand; HUMPHREY, Marty A.; LEWIS, Michael J.; NGUYEN-TUONG, Anh; KARPOVICH, John F.; MORGAN, Mark M.; ; FERRARI, Adam J. From legion to avaki: the persistence of vision. In: BERMAN, Fran; FOX, Geoffrey C.; HEY, Anthony J. G. (Eds.). **Grid computing**: making the global infrastructure a reality. [S.l.]: John Wiley & Sons, 2003. p.265–298.

[GRI06] GRIPHYN – grid physics network. [S.l.: s.n.], 2006. (`http://www.griphyn.org/`).

[IVD06] IVDGL – international virtual data grid laboratory. [S.l.: s.n.], 2006. (`http://www.ivdgl.org/`).

[KRA02] KRAUTER, Klaus; BUYYA, Rajkumar; MAHESWARAN, Muthucumaru. A taxonomy and survey of grid resource management systems for distributed computing. **Software – Practice and Experience**, v.32, n.2, p.135–164, 2002.

[LAF02] LAFORENZA, Domenico. Grid programming: some indications where we are headed. **Parallel Computing**, v.28, n.12, p.1733–1752, December 2002.

[MAN05] MANDRICHENKO, I.; ALLCOCK, W.; T.PERELMUTOV. GridFTP v2 protocol description. [S.l.: s.n.], 2005. (GFD-R-P.047. Disponível em `http://www.ggf.org/documents/GFD.47.pdf`).

[MIC06] MICROSYSTEMS, Sun. Java web services homepage. [S.l.: s.n.], 2006. (`http://java.sun.com/webservices/`).

[NAB03] NABRZYSKI, Jarek; SCHOPF, Jennifer M.; WEGLARZ, Jan. **Grid resource management**: state of the art and future trends. [S.l.]: Kluwer Academic Publishers, 2003. p.598.

[OUR06] OURGRID. [S.l.: s.n.], 2006. (`http://www.ourgrid.org/`).

[PAP03] PAPAZOGLOU, M. P.; GEORGAKOPOULOS, D. Service-oriented computing. **Communications of the ACM**, v.46, n.10, p.25–28, October 2003.

[PPD06] PARTICLE physics data grid. [S.l.: s.n.], 2006. (`http://www.ppdg.net/`).

[ROU03] ROURE, David De; BAKER, Mark A.; JENNINGS, Nicholas R.; SHADBOLT, Nigel R. The evolution of the Grid. In: BERMAN, Francine; FOX, Geoffrey; HEY, Tony (Eds.). **Grid computing**: making the global infrastructure a reality. [S.l.]: Wiley & Sons, 2003. p.65–100.

[SAN05] SANCHES, José A. L.; VARGAS, Patrícia Kayser; DUTRA, I. C.; SANTOS COSTA, Vítor; GEYER, Cláudio F. R. Regs: user-level reliability in a grid environment. In: , 2005, Cardiff, UK. **Anais...** 2005.

[SAN03] SANDER, Volker; ALLCOCK, William; CONGDUC, Pham; MONGA, Inder; PADALA, Pradeep; TANA, Marco; TRAVOSTINO, Franco. **Networking issues of grid infrastructures**. [S.l.: s.n.], 2003. (Available at `http://forge.gridforum.org/projects/ggf-editor/document/Networking_Issues_of_Grid_Infrastructures/en/1/Networking_Issues_of_Grid_Infrastructures.pdf`). Document draft-ggf-ghpn-netissues-0, version 1.

[SOT05] SOTOMAYOR, Borja; CHILDERS, Lisa. **Globus toolkit 4**. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2005.

[STO01] STOCKINGER, Heinz; RAN, Omer F.; MOORE, Reagan; MERZKY, Andre. Data management for grid environments. In: , 2001, Amsterdam, The Netherlands. **Anais...** 2001.

[TER06] TERAGRID project. [S.l.: s.n.], 2006. (`http://www.teragrid.org`).

[DAT06] THE datagrid project. [S.l.: s.n.], 2006. (`http://eu-datagrid.web.cern.ch/eu-datagrid/`).

[EAS06] THE easygrid project's research, reference and resource library. [S.l.: s.n.], 2006. (`http://easygrid.ic.uff.br/`).

[GLO06] THE globus toolkit. [S.l.: s.n.], 2006. (`http://www.globus.org/toolkit/`).

[GRA06] THE grid application development software (GrADS) project. [S.l.: s.n.], 2006. (`http://www.hipersoft.rice.edu/grads/`).

[WS-04] THE WS-Resource Framework. [S.l.: s.n.], 2004. (`http://www.globus.org/wsrf/`).

[TUE02] TUECKE, Steven; CZAJKOWSKI, Karl; FOSTER, Ian; FREY, James; GRAHAM, Steven; KESSELMAN, Carl; VANDERBILT, Peter. Grid service specification. [S.l.]: Global Grid Forum, 2002. (Available at `http://www.gridforum.org/ogsi-wg/drafts/GS_Spec_draft03_2002-07-17.pdf`).

[VAR05] VARGAS, Patrícia Kayser; BARRETO, Marcos Ennes. Gerenciamento de recursos em ambiente de grade. In: SIQUEIRA, Marcos Ennes Barreto e Mozart Lemos de (Ed.). **5a. escola regional de alto desempenho (erad2005)**. [S.l.]: SBC/UNILASALLE/UFPel/UCPel/UCS, 2005. p.3–32.

[VAR05a] VARGAS, Patrícia Kayser; CASTRO DUTRA, Inês de; NASCIMENTO, Vinícius Dalto do; SANTOS, Lucas A. S.; SILVA, Luciano C. da; GEYER, Cláudio F.R.; SCHULZE, Bruno. Hierarchical submission in a grid environment. In: , 2005, Grenoble, France. **Anais...** 2005.

[WEL03] WELCH, Von; SIEBENLIST, Frank; FOSTER, Ian; BRESNAHAN, John; CZAJKOWSKI, Karl; GAWOR, Jarek; KESSELMAN, Carl; MEDER, Sam; PEARLMAN, Laura; TUECKE, Steven. Security for grid services. In: , 2003, Seattle, Washington. **Anais...** 2003.

[YUR06] YURKEWICZ, Katie. Feature: a decade of globus in science. [S.l.: s.n.], 2006.