1
Gerência de Recursos em Ambientes de Grade

Patrícia Kayser Vargas[1] *(kayser@unilasalle.edu.br)*
Marcos Ennes Barreto[2] *(barreto@unilasalle.edu.br)*

Resumo:

A computação em grade ou *grid computing* é uma proposta promissora para resolver as crescentes demandas da computação paralela e distribuída por transparência, capacidade computacional e integração de recursos e serviços. A pesquisa em computação em grade visa desenvolver uma infra-estrutura de hardware e software que forneça o acesso consistente e pervasivo aos recursos computacionais distribuídos geograficamente em diferentes domínios administrativos.

Este tutorial apresenta uma introdução à computação em grade e discute alguns os principais pontos que representam o estado da arte em gerenciamento de recursos e de dados. Os recursos disponíveis em um ambiente de grade são heterogêneos, podendo compreender máquinas individuais, clusters de computadores, equipamentos específicos, tais como sensores meteorológicos; quanto bases de dados. Entre as tarefas envolvidas no gerenciamento de recursos serão discutidos os principais problemas, técnicas e propostas de protocolos e ferramentas para tratar a descoberta, a alocação e o escalonamento em grades. Adicionalmente, serão discutidos alguns tópicos relacionados ao gerenciamento de dados em ambientes de grade.

[1]Doutoranda da COPPE/Sistemas, UFRJ. Professora do UniLaSalle – Canoas, RS. Apoio CNPq.
[2]Doutorando do Instituto de Informática, UFRGS. Professor do UniLaSalle – Canoas, RS.

1.1. Introdução

Um sistema distribuído pode ser definido como "uma coleção de computadores independentes que parecem ao usuário como um único computador"[TAN 95]. A utilização de forma cooperativa e transparente de recursos distribuídos geograficamente considerando-os como um único e poderoso computador é algo desejado há bastante tempo, e que recentemente se mostra viável tecnologicamente através principalmente das pesquisas em computação em grade.

O termo Computação em Grade ou *Grid computing* [FOS 98, FOS 2001] foi estabelecido no final dos anos 90 e denota uma proposta de infra-estrutura computacional distribuída. A computação em grade é uma proposta promissora para resolver as crescentes demandas da computação paralela e distribuída por transparência, capacidade computacional e integração de recursos e serviços.

Uma infra-estrutura de grade deve prover, de forma global e transparente, os recursos heterogêneos e geograficamente distribuídos requisitados por aplicações de grande demanda computacional (CPU e/ou E/S), tais como aplicações em física de altas energias (HEP) ou em sequenciamento de genomas. Esta infra-estrutura pode ligar e unificar globalmente diversos recursos remotos, abrangendo de sensores meteorológicos a dados de estoque, de supercomputadores paralelos a organizadores digitais pessoais. Deste modo, é dito que deve prover "serviços pervasivos para todos os usuários que precisarem deles", levando ao conceito de computação pervasiva (*pervasive computing* ou *ubiquitous computing*[3]).

Uma das tarefas a serem tratadas pela infra-estrutura de grade é o gerenciamento de recursos computacionais e o gerenciamento de dados, estejam eles armazenados em arquivos ou em bases de dados. Vários trabalhos vêm sendo propostos para tratar o gerenciamento de recursos e aplicações no ambiente de grade [ROU 2003, BER 2003, THA 2003] Este texto apresenta alguns destes trabalhos. Entre as tarefas envolvidas no gerenciamento de recursos serão discutidos os principais problemas, técnicas e propostas de protocolos e ferramentas para tratar a descoberta, a alocação e o escalonamento em grades.

O restante deste texto encontra-se organizado do seguinte modo. Na seção 1.2. são apresentados conceitos e histórico de trabalhos relacionados a computação em grade. Depois, são apresentados diversos aspectos sobre gerenciamento de recursos, monitoração e gerenciamento de dados, respectivamente, nas seções 1.3., 1.4. e 1.5.. As considerações finais são apresentadas na seção 1.6..

1.2. Computação em grade

Um número crescente de grupos de pesquisa vêm trabalhando na área de computação distribuída em grandes distâncias (*wide-area distributed computing*). Estes grupos têm implementado *middlewares*, bibliotecas e ferramentas que permitem o uso coope-

[3]Os termos "pervasivo" e "pervasiva" são utilizados em alguns textos sobre computação sem fio e em grade, embora a tradução mais correta para o português talvez fosse difundido(a) ou ubíquo(a), isto é, algo que está ao mesmo tempo em toda a parte. O objetivo dos pesquisadores neste contexto é criar um sistema que esteja integrado ao ambiente computacional, estando totalmente conectado e constantemente disponível, bem como sendo intuitivo e realmente portável.

rativo de recursos distribuídos geograficamente. Esta tentativa de suprir as demandas crescentes por transparência, desempenho e agregação de capacidade de processamento para computação paralela e distribuída tem sido chamada por diversos nomes, tais como metacomputação (*metacomputing*), *seamless scalable computing*, *global computing* e, mais recentemente, computação em grade (*Grid Computing*) [BAK 2000, ROU 2003].

Segundo Foster *et al*. [FOS 2001], o termo computação em grade foi estabelecido no meio da década de 90 para denotar "uma proposta de infra-estrutura computacional distribuída para engenharia e ciências avançadas". Uma infra-estrutura de computação em grade deve prover acesso a recursos heterogêneos para aplicações com alta demanda por ciclos computacionais e/ou por dados, como por exemplo, aplicações em física de altas energias (*high energy physics* ou HEP) ou sequenciamento de genomas.

Baker *et al*. [BAK 2000] consideram que a popularização da Internet e a disponibilidade de computadores poderosos e redes de alta velocidade a baixo custo fornecem a oportunidade tecnológica de usar as redes de computadores como um recurso computacional unificado.

Outra definição para grade é apresentada em Krauter *et al*. [KRA 2002]: "uma grade é um sistema computacional de rede que pode escalar para ambientes do tamanho da Internet com máquinas distribuídas através de múltiplas organizações e domínios administrativos. Neste contexto, um sistema computacional de rede distribuído é um computador virtual formado por um conjunto de máquinas heterogêneas ligadas por uma rede que concordam em compartilhar seus recursos locais com os outros."

Um ambiente de grade ideal deveria prover acesso aos recursos disponíveis de forma homogênea de tal modo que descontinuidades físicas, tais como diferenças entre plataformas, protocolos de rede e bordas administrativas, se tornassem completamente transparentes.

Pode-se então dizer que uma grade é uma coleção de recursos computacionais heterogêneos, geograficamente espalhados, e pertencentes a diferentes domínios administrativos. Já um ambiente de grade é uma infra-estrutura que deve fornecer facilidades para que aplicações de diferentes usuários possam acessar estes recursos de forma transparente.

Os três principais aspectos que caracterizam uma grade computacional são heterogeneidade, escalabilidade e dinamicidade [BAK 2000]. A *heterogeneidade* é uma das características mais evidentes, já que uma 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.

Quanto a *escalabilidade*, uma grade pode crescer de poucos recursos para milhões. Isto levanta o problema da potencial degradação do desempenho a 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 a latência.

Um ambiente de grade deve considerar os aspectos de *dinamicidade* ou *adaptabilidade*, já que 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.

1.2.1. Computação em grade e sistemas distribuídos

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*) [FOS 2001, LAF 2002]. Devido ao seu foco em compartilhamentos inter-organizacionais e dinâmicos, as tecnologias de grade complementam-se, ao invés de competir, com as tecnologias de computação distribuída existentes [FOS 2001].

Németh and Sunderam [NEM 2002] apresentaram uma definição formal sobre o que um ambiente de grade deveria prover. Eles argumentam que uma grade não é apenas uma modificação de sistemas distribuídos "convencionais", mas sim que há uma diferença semântica. Eles apresentam uma comparação entre sistemas distribuídos e grade que transcrevemos na tabela 1.1. Uma grade pode apresentar recursos heterogêneos, incluindo não apenas nodos computacionais como também, por exemplo, sensores e detectores. Em teoria, o usuário tem pouco conhecimento sobre cada recurso individual, principalmente devido à restrições administrativas e à grande quantidade de recursos disponíveis. Dessa forma, o usuário deve sempre acessar a grade (o conjunto de recursos) ao invés de recursos individuais. Finalmente, enquanto sistemas distribuídos convencionais tendem a serem estáticos, exceto por falhas e manutenção, grades são dinâmicas por definição.

Tabela 1.1: Comparação entre sistemas distribuídos e grade [NEM 2002].

	Sistemas Distribuídos Convencionais	**Grades**
1	um conjunto virtual de nodos computacionais	um conjunto virtual de recursos
2	o usuário tem acesso (credencial) a todos os nodos do conjunto	o usuário tem acesso ao conjunto mas não aos domínios individuais
3	o acesso a um nodo significa acesso a todos os recursos no nodo	o acesso a um recurso pode ser restrito
4	o usuário está ciente das potencialidades (*capabilities) e características dos nodos*	o usuário possui pouco conhecimento sobre cada recurso
5	nodos pertencem a uma única organização (*trust domain)*	recursos estão espalhados por múltiplas organizações
6	elementos em um conjunto em torno de 10-100, mais ou menos estáticos	elementos em um conjunto em torno de 1000-10000, dinâmicos

1.2.2. Aplicações para grades computacionais

Foster and Kesselman [FOS 98a] identificam cinco grandes classes de aplicações para ambientes de grade:

- *Distributed supercomputing applications*: utilizam a grade para agregar recursos computacionais significativos de modo a tratar problemas que não podem ser resolvidos em um sistema isolado. Estas aplicações resolvem problemas grandes, tais como simulação de processos físicos complexos, que requerem grandes quantidades de recursos, tais como processador e memória.

- *High-throughput computing*: usam os recursos da grade para escalonar um grande número de tarefas independentes ou fracamente acopladas, com o objetivo de aproveitar ciclos ociosos de processador. O sistema Condor [THA 2003], por exemplo, vem tratando deste tipo de aplicação já há algum tempo. Outros exemplos compreendem simulação molecular de cristal líquido e problemas bioestatísticos resolvidos com programação em lógica indutiva.

- *On-demand computing applications*: usam a capacidade da grade para atender uma demanda por recursos que por uma questão de conveniência ou de custo não são mantidos localmente. Por exemplo, o processamento de dados meteorológicos pode usar supercomputadores obtidos dinamicamente para executar um algoritmo de detecção de nuvens.

- *Data-intensive computing applications*: possui como foco a síntese de novas informações a partir de dados que são mantidos em repositórios, bibliotecas digitais e bancos de dados distribuídos geograficamente. Freqüentemente este processo é intensivo em termos de computação e comunicação, como é esperado nos futuros experimentos de HEP.

- *Collaborative computing applications*: estão preocupadas principalmente em permitir e melhorar as interações humanas. Muitas aplicações colaborativas buscam permitir o compartilhamento de recursos computacionais, tais como arquivos de dados e simulações. Por exemplo, o sistema CAVE5D [CAV 2004] suporta a exploração colaborativa e remota de grandes conjuntos de dados geofísicos e dos modelos que os geraram.

1.2.3. Histórico

Roure *et al*. [ROU 2003] identificam 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 composta 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 foco em *middleware* para suporte a dados e computação em larga escala. Caracterizada principalmente por projetos que tratam de infraestrutura, serviços básicos, aplicações específicas e portais de domínio; e

- *terceira geração*: é a geração corrente onde a ênfase muda para questões de colaboração global distribuída, abordagem orientada a serviços e camadas de informação.

Dois projetos representativos da primeira geração são FAFNER e I-WAY.

FAFNER (*Factoring via Network-Enabled Recursion*) [FAF 2000][FAF 2000a] foi criado através de um consórcio a fim de realizar a fatoração RSA130 usando uma técnica numérica denominada *Number Field Sieve*. O consórcio produziu uma interface WEB para a aplicação do *Number Field Sieve*. Colaboradores usavam um formulário Web para invocar no servidor um conjunto de CGI (*Common Gateway Interface*) scripts escritos em Perl.

Segundo Roure *et al*. [ROU 2003], três fatores contribuíram para o sucesso da abordagem proposta no projeto FAFNER: (1) a implementação da aplicação permitia que mesmo estações de trabalho com 4Mb de memória realizassem trabalho útil usando limites menores e uma fatoração menor; (2) FAFNER suportava registro anônimo; (3) vários sítios foram recrutados formando uma rede hierárquica de servidores Web de RSA130, o que reduziu o gargalo potencial de administração e permitiu que a fatoração acontecesse.

O I-WAY (*The Information Wide Area Year*) [DEF 96] foi concebido no início de 1995 como uma rede experimental de alto desempenho interligando vários computadores de alto desempenho e ambientes avançados de visualização. A idéia não era a construção de uma rede mas a integração de redes já existentes. O esforço de aproximadamente um ano concretizou a interligação de dezessete diferentes sítios nos Estados Unidos conectados por dez redes com bandas e protocolos variáveis, usando diferentes tecnologias de roteamento e *switching*.

Alguns dos principais pontos explorados no I-WAY foram: (a) uso de uma rede IP/ATM para supercomputação em larga escala; (b) estabelecimento de um *framework* para a construção de uma infra-estrutura nacional (Estados Unidos) de pesquisa, estabelecendo parcerias entre provedores (*carriers*), vendedores e pesquisadores; (c) construção da primeira infra-estrutura através dos EUA para suportar projetos colaborativos em larga escala; (d) desenvolvimento de um protótipo de ambiente de software e um escalonador para larga escala para facilitar o uso de recursos distribuídos sem necessitar o conhecimento de suas localizações e configuração, e com alto grau de segurança.

Estes dois projetos diferem em várias formas, mas ambos tentam superar uma série de obstáculos similares, incluindo comunicação, gerenciamento de recursos e manipulação de dados remotos, para serem capazes de trabalhar eficientemente e efetivamente. Tanto FAFNER quanto I-WAY buscaram a obtenção de ambientes de metacomputação pela integração de recursos. Os clientes FAFNER eram tipicamente computadores de usuários finais enquanto o I-WAY buscava integrar recursos em centros de supercomputação. Os dois projetos também se diferenciavam na questão das aplicações: o primeiro foi feito para uma aplicação em particular que era trivialmente paralela e que não dependia de uma interconexão rápida; por outro lado, o I-WAY foi projetado para tratar de um conjunto de aplicações de alto desempenho que tipicamente necessitam de interconexão rápida e recursos poderosos.

Ambos os projetos de certa forma tinham problemas de escalabilidade: o FAFNER dependia bastante da intervenção humana para distribuir e coletar resultados; o I-WAY estava limitado pelo projeto dos componentes que faziam parte dos I-POP e I-Soft[4]. Cada projeto estava na vanguarda da metacomputação e ajudou a facilitar o desenvolvimento de vários projetos da segunda geração. O FAFNER foi o precursor de projetos tais como SETI@home (*The Search for Extraterrestrial Intelligence at Home*) [SET 2004] e Distributed.Net [DIS 2004]. Já o I-WAY foi o precursor de projetos tais como o Globus e o Legion.

1.3. Gerenciamento de Recursos

Uma das parte centrais de um sistema distribuído é o gerenciador de recursos (RMS ou *Resource Management System*). Em uma grade, um gerenciador de recursos

[4]"Pontos de presença", como eram chamados cada um dos dezessete sítios usados no projeto.

deve gerenciar um conjunto de recursos que estão disponíveis e que podem incluir recursos de diferentes provedores. Os recursos gerenciados são principalmente processadores, banda de rede e discos de armazenamento. O ambiente de grade introduz cinco problemas ao gerenciamento de recursos [CZA 98]:

1. autonomia nos sítios: os recursos tipicamente pertencem e são operados por diferentes organizações, em domínios administrativos diferentes;
2. base heterogênea: sítios podem usar diferentes RMS locais ou o mesmo sistema com diferentes configurações;
3. facilidade de extensão de políticas: um RMS deve suportar o desenvolvimento de novos mecanismos de gerenciamento específicos para um determinado domínio de aplicação, sem exigir mudanças no código instalado nos sítios participantes;
4. co-alocação: algumas aplicações possuem requisitos relativos aos recursos que podem ser satisfeitos apenas pelo uso simultâneo de recursos em diferentes sítios;
5. controle *online*: um RMS deve suportar negociação para adaptar os requisitos da aplicação à disponibilidade de recursos.

Ambientes de grade são compostos por vários domínios administrativos. Normalmente, cada domínio possui o seu escalonador privado, que trabalha isolado dos demais. É necessário um gerenciador de recursos que conheça a grade para permitir que estes escalonadores isolados trabalhem junto para aproveitar o potencial da grade. Vários trabalhos na literatura apresentam sistemas que realizam escalonamento em ambiente de grade. Alguns destes trabalhos são apresentados nas próximas subseções.

1.3.1. Legion

O projeto Legion, da University of Virginia, iniciou em 1993. O principal resultado deste projeto é o sistema Legion [GRI 99, CHA 99] que é um ambiente orientado a objetos para ambientes de grade. Atualmente o Legion está sendo comercializado pela Avaki [AVA 2004].

O Legion é um *middleware* composto de vários componentes, tais como gerenciamento de recursos, gerenciamento de dados e segurança. Ele provê a ilusão de uma única máquina virtual com segurança contra acesso indevido e processamento distribuído. O Legion usa sistemas operacionais, ferramentas de gerenciamento de recursos e mecanismos de segurança existentes para implementar serviços de mais alto nível. Proposta ambiciosa, se propõe a atingir diversos objetivos, dentre os quais: autonomia do site, suporte a heterogeneidade, extensibilidade, facilidade de uso, processamento paralelo para atingir desempenho, tolerância a falhas e escalabilidade.

O modelo do Legion é baseado em objetos, onde objetos ativos se comunicam via invocação remota de métodos. A figura 1.1 apresenta a hierarquia dos objetos básicos que compõem o Legion (*Legion Core Object Hierarchy*). `LegionClass` é a classe mais geral a partir da qual todas as demais classes, inclusive as de usuário, são estendidas. Uma `HostClass` descreve características de uma máquina, enquanto uma VaultClass é uma abstração de uma unidade de armazenamento em geral. Estes dois últimos tipos de objetos têm um importante papel no gerenciamento de recursos conforme será discutido a seguir.

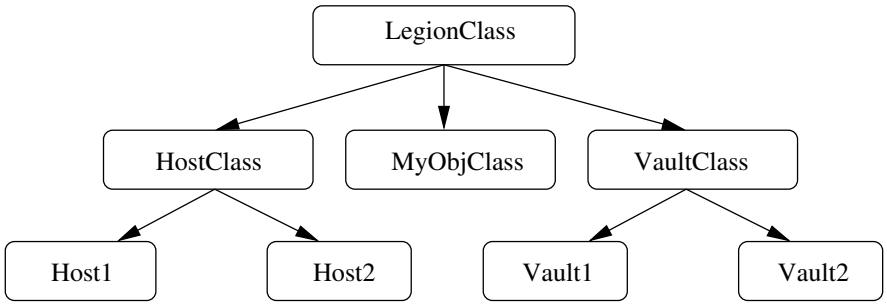

Figura 1.1: Legion: hierarquia de objetos básicos [GRI 99].

Cada classe possui um `ClassManager` associado a ela. Se um `ClassManager` falha, seu `ClassManager` de mais alto nível detecta a perda e pode reiniciar (caso haja replicação). Para implementar esta dependência hierárquica é utilizado um mecanismo de notificação baseado em eventos.

O fato de usar uma abordagem orientada a objetos facilita a extensibilidade. Como os objetos são diferenciados apenas pela sua interface (métodos publicados), é possível ter mais de uma implementação para a mesma interface.

Um dos aspectos fundamentais implementados no ambiente Legion é o gerenciamento de recursos. Na figura 1.2 é possível identificar os componentes principais envolvidos no gerenciamento de recursos:

- `Host`: periodicamente atualiza seu estado local. Quando um host recebe requisição de reserva deve garantir que um `vault` está disponível, que existam recursos disponíveis suficientes e que a política local de alocação permita a instanciação do objeto;

- `Vault`: não contém informações dinâmicas de estado, apenas participa no início do escalonamento para verificar compatibilidade com `hosts`;

- `Collection`: é um repositório para informações que descreve o estado dos recursos que compõem o sistema; ou seja, é um banco de dados passivo de informações estáticas. No collection, cada registro é armazenado como um conjunto de atributos de objeto. Para obter informações deste repositório, realiza-se consultas com uma linguagem específica que possui uma sintaxe bastante intuitiva. Um exemplo de consulta seria:

```
match($host_os_name, "IRIX") and match($host_os_name, "5\..*")
```

- `Scheduler`: realiza o mapeamento de objetos para recursos. Para tanto, ele deve: obter informações do `collection`; computar o mapeamento das instâncias de objetos a recursos; enviar o mapeamento (escalonador - escalonamento) para o `enactor`.

- `Enactor`: cada entrada do escalonamento recebido é "executada" por ele. Se todos os mapeamentos do escalonamento recebido (*master schedule*) tiverem sucesso, então o escalonamento está completo. Senão, seleciona uma alternativa (*variant schedule*) que contenha uma nova entrada para o mapeamento que falhou.

A seqüência das interações entre os módulos ilustrada na Figura 1.2 é a seguinte:

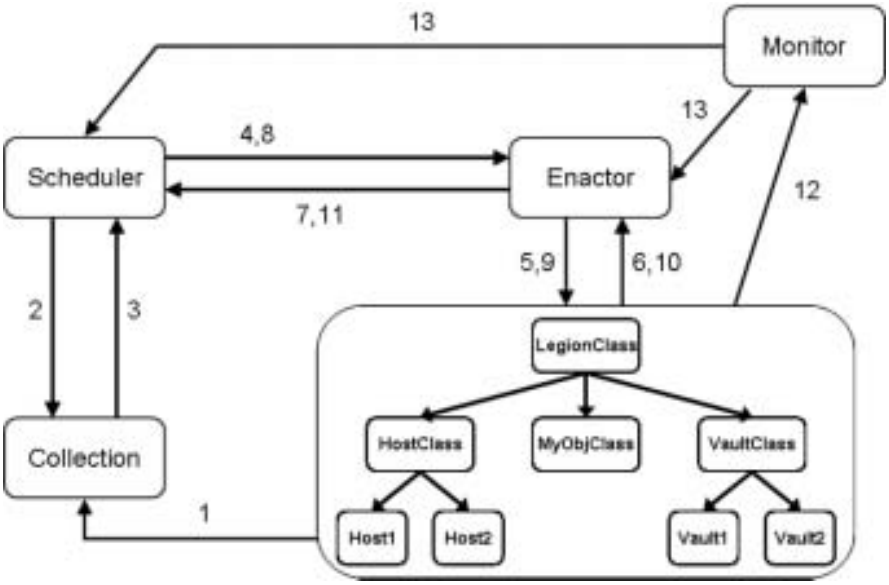

Figura 1.2: Legion: gerenciamento de recursos [GRI 99].

1. `Collection` é preenchido com informações descrevendo os recursos
2. O `Scheduler` consulta `Collection` e
3. baseado no resultado e conhecimento da aplicação, realiza o mapeamento dos objetos aos recursos. Este conhecimento específico da aplicação pode ser tanto implícito (*application-specific Scheduler*) ou pode ser obtido das classes da aplicação.
4. Este mapeamento é passado ao `Enactor`, o qual
5. invoca métodos nos `hosts` e `vaults` para
6. obter reservas dos recursos nomeados no mapeamento.
7. Após obter as reservas, o `Enactor` contacta o `Scheduler` para confirmar o escalonamento, e
8. após receber a aprovação do `Scheduler`,
9. tenta instanciar os objetos através de chamadas de função nas classes de objetos apropriadas.
10. As classes dos objetos reportam códigos de sucesso/falha, e
11. o `Enactor` retorna o resultado para o `Scheduler`.
12. Se, durante a execução, um recurso decidir que um objeto precisa ser migrado, ele realiza uma chamada ao `Monitor`,
13. o qual notifica o `Scheduler` e o `Enactor` que deve ser realizado um novo escalonamento.

1.3.2. Globus

O projeto Globus é um dos projetos mais referenciados na literatura, sendo conduzido principalmente no Argonne National Laboratory, na University of Southern California e na University of Chicago.

O Globus Toolkit [FOS 98b, SAN 2003] é um software livre que fornece diversos serviços que podem ser usados para a construção de sistemas computacionais para grade. O *toolkit* consiste de um conjunto de componentes que implementam serviços básicos, tais como segurança, alocação de recursos, gerenciamento de dados e comunicação. A versão 1.0 foi lançada em 1998. A versão 2.0 (GT2), de 2002, possui seu projeto fortemente relacionada à arquitetura proposta por Foster *et al*. em [FOS 2001]. O projeto do Globus Toolkit versão 3 (GT3) é baseado em *grid services*, que são muito parecidos com *web services*. O GT3 implementa conceitos da OGSI *Open Grid Service Infrastructure* [FOS 2002]. Em 2004 foi lançada a a especificação da *WS-Resource Framework* (WSRF), que propôs mudanças na OGSI, usando conceitos de *web services*. O WSRF está sendo usado para construção do GT4 cujo lançamento da primeira versão estável está prevista para janeiro de 2005.

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.

Considerando a arquitetura de referência proposta por Foster *et al*. [FOS 2001], tem-se os seguintes componentes em cada camada:

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

- *conectividade*: com relação à comunicação, propõe o uso de protocolos padrões da internet (p.ex. IP). Com relação a segurança, existe o *Grid Security Infrastructure (GSI)* que realiza autenticação, autorização e mecanismos de proteção de mensagens de maneira uniforme. O GSI permite *single sign-on* e delegação, utilizando tecnologia de chave pública, SSL, X.509 e GSS-API.

- *recurso*: esta camada contém os seguintes serviços: *Grid Resource Allocation Management (GRAM)*, que permite alocação remota, reserva, monitoração e controle dos recursos computacionais; *GridFTP* (extensão do protocolo FTP), que permite acesso de alto desempenho aos dados; *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*: nesta camada existiriam serviços como *checkpoint* distribuído, gerenciamento de tarefas e replicação que não estão contemplados na distribuição do Globus Toolkit.

Assim, os principais componentes que compõem o Globus Toolkit são o GSI, o GRAM, o GridFTP e GRIS. No entanto, é importante ressaltar que esta visão modular é característica do GT2. O GT3 apresenta uma abordagem de serviços, isto é, os vários

componentes do *toolkit* são vistos como serviços que podem ser disponibilizados à clientes, não podendo ser encaixados nesta estrutura de camadas.

Grid Security Infrastructure (GSI) A questão de segurança é um fator vital para determinar o sucesso de um ambiente de computação em grade, uma vez que nenhum administrador se sentiria seguro em permitir que suas máquinas pertencessem a uma grade sem garantias mínimas de segurança. Dentre estas garantias mínimas tem-se a necessidade de garantir a autenticação (estabelecimento da identidade do usuário) e a autorização (estabelecimento dos direitos que um usuário possuem em relação aos diversos recursos da grade). Além disso, é necessário garantir a segurança em relação às mensagens, isto é, garantir proteção, integridade e confidencialidade.

Deste modo, as principais motivações para a inclusão de um módulo que trate de segurança no Globus Toolkit são: (a) a necessidade de comunicação segura (autenticada e talvez confidencial) entre elementos da grade computacional; (b) a necessidade de suportar segurança entre os limites organizacionais, e portanto, evitando um sistema de segurança de gerência centralizada; e (c) a necessidade de suportar *single sign-on* para os usuário da grade, incluindo delegação de credenciais para computação que envolva múltiplos recursos e/ou sítios.

O módulo encarregado de suprir esta demanda por segurança é GSI (*Grid Security Infrastructure*) [WEL 2003] que utiliza certificados de autoridade (*certificate authority ou CA*). Um CA é uma "autoridade" na rede que trata e gerencia credenciais de segurança e chaves públicas para criptografia de mensagens. Como parte da *Public Key Infrastructure* (PKI), um CA verifica se as informações prestadas por um requisitante, através de um registro de autoridade, está de acordo a fim de disponibilizar um certificado digital.

O funcionamento de uma execução no ambiente de grade usando GSI segue as seguintes etapas: **(1)** obtenção do *proxy credential* através da assinatura única para a grade (*single sign-on via grid-id*) ou através da recuperação em um repositório; **(2)** de posse da credencial é solicitada a criação de processos remotos; **(3)** o servidor GRAM com GSI se encarrega de autorizar, mapear o ID de grade para o ID local, criar os processos e, quando for o caso, criar novas credenciais; **(3)** a requisição de um processo no sítio A para acessar um arquivo no sítio B necessariamente passa por um processo de autenticação mútua e, portanto, o servidor de FTP tem que estar preparado para reconhecer o GSI.

GridFTP protocol GridFTP [ALL 2002, ALL 2002a, The 2000] 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 (nos IETF RFCs) bem como novas características foram adicionadas para atender aos requisitos dos atuais projetos de *data grid*. Das funcionalidades mais comumente utilizadas tem-se principalmente o envio e recebimento de arquivo (get, put, etc). Em outras características padrão pouco utilizadas tem-se listagem estendida de diretórios, reinício simples de transmissão e ligação com GSS. Dentre as novas características adicionadas, destaca-se: canais de dados paralelos (*stripped/parallel data channels*), envio parcial de arquivos, configuração manual e automática do buffer TCP, monitoração do progresso e extensão da funcionalidade de reinício.

Globus Toolkit 3 (GT3) O núcleo da infra-estrutura do Globus Toolkit 3 (GT3) é construído em cima das primitivas e protocolos da *Open Grid Service Architecture* (OGSA) [TUE 2002], que é uma implementação *open source* da *Open Grid Service Infrastructure* (OGSI) [FOS 2002].

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* – 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 é especificado e pode ser usado para construir e compor novos serviços.

A arquitetura do núcleo do GT3 pode ser visualizada na figura 1.3.

Figura 1.3: Globus: arquitetura do Globus Toolkit 3.

A implementação da `infra-estrutura de segurança` (*security infrastructure*) provê 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, um 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.

Globus Toolkit 4 (GT4) O grupo de pesquisa Globus e a IBM propuseram o WS-Resource Framework [WSR 2004]. Eles lançaram uma primeira arquitetura e especificação em janeiro de 2004, em co-autoria com HP, SAP, Akamai, TIBCO e Sonic. O WS-Resource Framework (WSRF) é uma extensão da OGSI composto com um conjunto de seis especificações de *web services*.

Existe atualmente uma versão beta disponível e foi anunciado que por volta de janeiro de 2005 uma versão estável seria lançada.

1.3.3. Condor e Condor-G

Condor High Throughput Computing System [THA 2003, TAN 2002, WRI 2001], ou simplesmente Condor, é um sistema de gerenciamento de recursos. Ele pode ser usado para gerenciar um conjunto de nodos dedicados assim como para utilizar ciclos ociosos de CPU em estações de trabalho. O uso do poder computacional de estações de trabalho ociosas é o principal diferencial entre o Condor e os sistemas gerenciadores de recursos (RMSs) tradicionais.

O Condor é composto por uma coleção de diferentes *daemons* [WRI 2001]. A figura 1.4 apresenta uma visão esquemática de um Condor *pool* com os seus *daemons*. O Condor exerce um controle administrativo sobre o *pool*. Um *pool* é um conjunto de recursos que se reportam para um único *daemon* chamado `collector`. O `collector` é um repositório central de informações. Quase todos os demais *daemons* enviam atualizações periódicas para ele. Cada atualização está na forma de *ClassAd*, uma estrutura de dados consistindo de um conjunto de atributos que descrevem uma entidade específica do sistema. A máquina onde o `collector` executa é chamada de gerenciador central (*central manager*).

No gerenciador central também executa o `negotiator`, o qual periodicamente realiza um ciclo de negociação. Este ciclo é um processo de *matchmaking*, onde o `negotiator` tenta realizar o "casamento" (*match*) entre os vários ClassAds (requisição de recursos e oferta de recursos). Um vez que um casamento é realizado, ambas as partes são notificadas e são responsáveis por agir em relação a este casamento.

O *daemon* `startd` executa em todas as máquinas do *pool*. Ele monitora as condições do recurso onde ele está executando, publica ClassAds com oferta de recurso e é responsável por garantir as políticas de uso estabelecidas pelo dono do recurso com relação ao início, suspensão e remoção de tarefas. Por exemplo, um dono de recurso pode especificar quais usuários ou tarefas devem ter preferência. Qualquer máquina executando o `startd` pode ser referenciada como uma máquina de execução (*execute machine*), uma vez que ela é capaz de executar tarefas do Condor.

Um usuário submete as suas tarefas ao *daemon* `schedd`. Este *daemon* mantém uma lista persistente de tarefas, publica ClassAds com requisição de recursos e negocia recursos. Após ter recebido um casamento (*match*) para uma dada tarefa, o `schedd` entra em um protocolo de solicitação (*claiming protocol*) diretamente com o `startd`. Através deste protocolo, o `schedd` apresenta um ClassAd descrevendo uma tarefa para o `startd` e requisita o controle temporário sobre o recurso. Uma vez que ele tenha solicitado um dado recurso, o `schedd` realiza o seu próprio escalonamento local para decidir que tarefas executar. Qualquer máquina executando um `schedd` pode ser re-

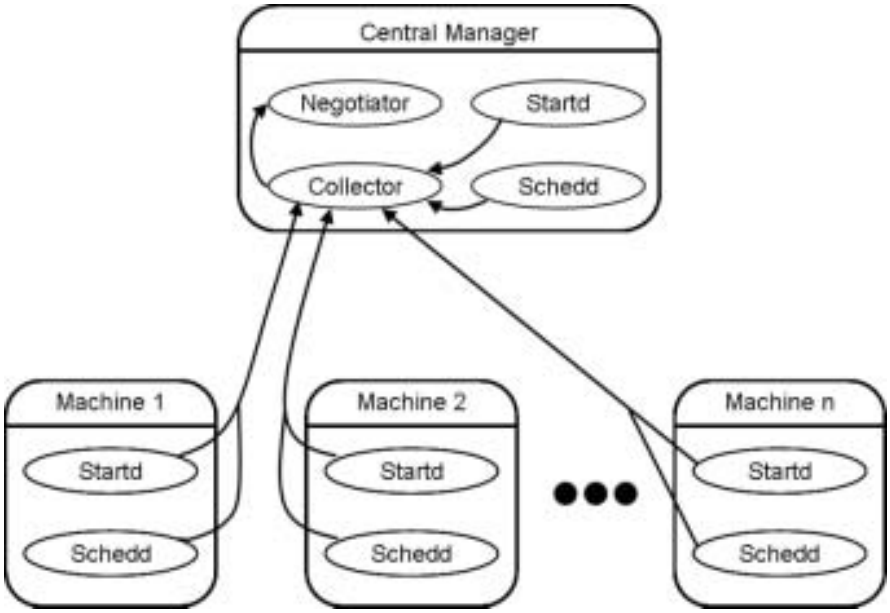

Figura 1.4: Arquitetura de um Condor *pool* com os seus *daemons* [WRI 2001].

ferenciada com uma máquina de submissão (*submit machine*), uma vez que os usuários podem submeter tarefas Condor a partir desta máquina.

Quando um `schedd` inicia uma tarefa, é disparado um processo `shadow` na máquina de submissão e o processo `startd` dispara um processo `starter` na máquina de execução correspondente. O `shadow` realiza requisição para transferência de arquivos, faz log das tarefa e reporta estatísticas. O `starter` configura o ambiente de execução e monitora a tarefa.

O Condor permite que o usuário execute suas tarefas em diferentes ambientes de execução chamados universos. Existem vários, mas os dois principais universos suportados são os seguintes [TAN 2002]: (a) *Vanilla*: usado para executar programas não paralelos (qualquer programa que funcione fora do Condor irá executar neste universo); e (b) *Standard*: uma tarefa neste ambiente pode realizar chamadas ao sistema operacional e pode utilizar os mecanismos de migração e *checkpoint* (neste caso as tarefas devem ser religadas com a biblioteca Condor).

O *checkpointing* que é realizado no universo *Standard* consiste em guardar o estado corrente de uma tarefa em execução para permitir que uma tarefa seja reiniciada deste ponto em caso de falha ou preempção. O Condor é configurado para realizar periodicamente o *checkpoint* das tarefas em uma máquina centralizada.

O Condor possui um mecanismo denominado *flocking* que permite que tarefas sejam escalonadas através de múltiplos Condor *pools* [THA 2003]. Atualmente, é implementado um mecanismo direto que apenas requer acordo entre um usuário e outra organização, mas que irá beneficiar apenas este usuário que toma a iniciativa. Uma tarefa em particular somente será direcionada para outro *pool* quando ela não puder ser executada no *pool* de submissão. Esta é uma característica útil, mas que não é suficiente para permitir que tarefas executem em um ambiente de grade, principalmente devido a problemas de segurança.

Condor-G[THA 2003, FRE 2002] é a parte de gerenciamento de tarefas do projeto Condor que permite que usuários acessem os recursos de uma grade. Ao invés de

utilizar os protocolos desenvolvidos para o Condor para inicializar a execução de uma tarefa em uma máquina remota, o Condor-G usa o Globus Toolkit para esta inicialização. Deste modo, aplicações podem ser submetidas para um recurso acessível através de uma interface Globus.

O Condor-G usa os protocolos para comunicação segura entre domínios e acesso padronizado para uma variedade de sistemas remotos de escalonamento através do Globus. As funcionalidades de submissão de tarefas, alocação de tarefas, recuperação de erros e criação de um ambiente amigável de execução vêm através do Condor.

A maior diferença entre o Condor *flocking* e o Condor-G é que o Condor-G permite operações entre domínios sobre recursos remotos que exigem autenticação e usa os protocolos padrão do Globus para prover acesso a recursos controlados por outros RMSs, ao invés de usar apenas os mecanismos de compartilhamento do Condor [FRE 2002].

1.3.4. MyGrid e OurGrid

O MyGrid [CIR 2001, PAR 2003] permite a execução de aplicações paralelas do tipo *bag-of-tasks* em toda e qualquer máquina que o usuário tenha acesso.

Uma aplicação do tipo *bag-of-tasks* é na qual as tarefas são independentes entre si, o que facilita a execução em plataformas geograficamente distribuídas. Um exemplo desse tipo de aplicação é o projeto SETI@home [SET 2004].

O escalonamento de tarefas independentes, embora mais simples do que em aplicações paralelas fortemente acopladas, é ainda difícil devido ao comportamento dinâmico e a intrínseca heterogeneidade de recursos exibida em ambientes de grade [PAR 2003]. É difícil obter boas informações sobre a grade como um todo assim como sobre as tarefas a fim de obter um bom escalonamento. Por isso, MyGrid propõe uma solução que não exige nenhum tipo de informação, usando o algoritmo de fila com replicação (*Workqueue with Replication* ou WQR).

O WQR é um algoritmo de escalonamento dinâmico que não utiliza informações de desempenho, sendo uma extensão da fila de trabalho (*workqueue*). O algoritmo *workqueue* funciona do seguinte modo: tarefas são escolhidas de forma arbitrária no grupo de tarefas e são enviadas para os processadores, a medida que estes se tornam disponíveis. Depois que a tarefa termina, o processador envia os resultados e o escalonador atribui uma nova tarefa para o processador. O problema com este algoritmo ocorre quando uma tarefa grande é alocada para uma máquina lenta próximo do final do escalonamento. Quando isto ocorre, o término da aplicação é atrasado até que a execução desta seja completada.

O algoritmo WQR usa replicação de tarefas para tratar da heterogeneidade de máquinas e tarefas, e também da variação dinâmica da disponibilidade de recursos devido a carga gerada por outros usuários na grade. Note que esta estratégia permite tratar apenas a heterogeneidade relacionada à capacidade computacional. O algoritmo WQR funciona como o algoritmo *workqueue* até que todas as tarefas sejam atribuídas. Neste momento, máquinas que já terminaram as suas tarefas vão executar réplicas de tarefas que já estão em execução. Tarefas são replicadas até que um número máximo de réplicas seja atingido (no MyGrid, o padrão é um). Esta replicação leva a perda de ciclos de CPU. Note que a replicação assume que as tarefas não possuem efeito colateral, podendo ser executadas mais de uma vez sempre produzindo os mesmos resultados finais – execução idempotente.

O MyGrid executa no nível do usuário. Existe uma máquina chamada *home machine* que coordena a execução da aplicação através do MyGrid. Assume-se que o usuário possui bom acesso a esta máquina, sendo freqüentemente a máquina de trabalho

do usuário. A *home machine* escalona as tarefas nas *grid machines* usando o algoritmo WQR. As *grid machines* não necessariamente compartilham o mesmo sistema de arquivos da *home machine*. Para acessar os recursos da grade, existe uma *Grid Machine Interface* que disponibiliza quatro serviços: (1) inicialização de tarefas nas *grid machines* (execução remota); (2) cancelamento de uma tarefa em execução; (3) transferência de arquivos da *grid machine* para a *home machine*; e (4) transferência de arquivos da *home machine* para a *grid machine*.

O OurGrid [AND 2003] estende o MyGrid permitindo a criação de comunidades de larga escala. O OurGrid pode ser definido como um sistema de compartilhamento de recursos baseado em tecnologias ponto-a-ponto (*peer-to-peer*). Os recursos são compartilhados de acordo com uma modelo de "rede de favores", na qual cada par (*peer*) prioriza a disponibilização de recursos para aqueles que possuem crédito em seu histórico de interações.

1.3.5. ISAM/EXEHDA

O projeto ISAM (Infra-estrutura de Suporte às Aplicações Móveis) [YAM 2003, YAM 2002, YAM 2002] propõe uma solução integrada, desde o desenvolvimento até a execução, para aplicações pervasivas de propósito geral. Estas aplicações são distribuídas, móveis, adaptativas e sensível ao contexto. Objetivando suportar a semântica *follow-me* (aplicações que "seguem" o usuário) para aplicações pervasivas, o *ISAM middleware* se preocupa em realizar gerenciamento de recursos em redes heterogêneas e multi-institutionais.

A arquitetura do ISAM é organizada em camadas com três níveis de abstração conforme representado na figura 1.5.

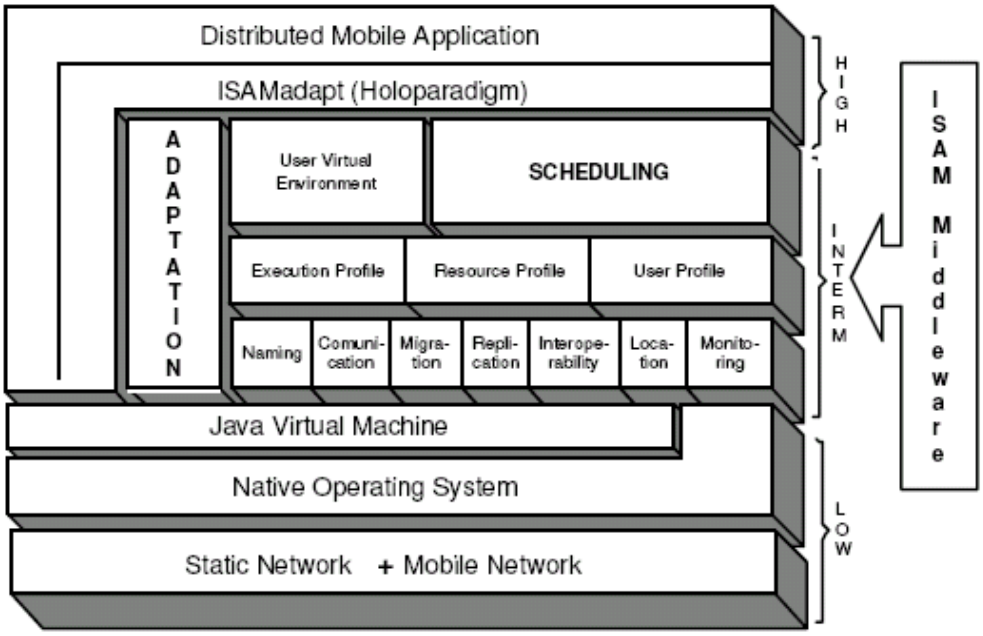

Figura 1.5: Arquitetura do ISAM [REA 2003].

No nível mais alto, o ISAM provê uma linguagem de programação denominada ISAMadapt, a qual permite que o programador expresse aspectos de adaptação. O ISA-

Madapt é construído com base no modelo de execução definido pelo Holoparadigma [BAR 2002], sendo compilado para código Java que acessa os serviços do *ISAM middleware*.

O nível intermediário é o *ISAM middleware*. Ele provê suporte para o ambiente virtual do usuário, um dos componentes chaves para o suporte à mobilidade do usuário. O escalonamento é um dos componentes principais relacionados ao gerenciamento da execução pervasiva.

O TiPS (*TiPS is a Probabilistic Scheduler*) [REA 2003, REA 2003a] é parte do componente de escalonamento. O TiPS provê uma estratégia de escalonamento baseada em sensores dinâmicos, o que inclui um modelo bayesiano do ambiente gerenciado.

O ambiente distribuído é gerenciado pelo ISAM em quatro níveis: (a) *hosts* são as máquinas; (b) *network segments* são conjuntos de *hosts*; (c) *computing cells* que delimitam os limites das instituições; e (d) *cell groups* são grupos de *computing cells*. As decisões de escalonamento são feitas considerando esta hierarquia.

Finalmente, o nível mais baixo do ISAM é o nível de infra-estrutura com os componentes físicos, o sistema operacional e a máquina virtual Java.

1.4. Monitoração

Ferramentas de monitoração são uma parte fundamental de uma infra-estrutura completa de grade. A monitoração da execução da aplicação e dos recursos é útil tanto do ponto de vista do sistema de grade quanto do usuário. A disponibilização de dados durante a execução permite que o sistema detecte falhas no sistema e tome atitudes para contornar o problema. Também permite que durante a execução a política de escalonamento se adapte a carga atual dos sistema. Já o usuário pode acompanhar a evolução da execução, permitindo inclusive que ele possa decidir iniciar a análise de dados parciais.

Um dos desafios da monitoração do estado da grade é a construção de um sistema escalável que consiga rapidamente detectar falhas. Além disso, é importante que ele permita tanto uma visão global como uma visão detalhada de subconjuntos de recursos. Por exemplo, com uma visão global pode ser possível detectar gargalos no sistema, enquanto que com a visão da interação entre um subconjunto de nodos, dispositivos de armazenamento e ligações de rede, é possível entender o porquê do gargalo.

Segundo Massie *et al*. [MAS 2004], os principais desafios de projeto de sistemas de monitoração distribuída incluem: (a) escalabilidade: conseguir se adaptar ao aumento do número de nodos; (b) robustez: sobreviver a falhas no sistema monitorado; (c) extensibilidade: permitir novos tipos de dados a serem monitorados; (d) capacidade de gerenciamento: ter baixo custo de gerenciamento, evitar configurações manuais; (e) portabilidade: ser portável para diferentes sistemas operacionais e tipos de CPU; (f) custo adicional (*overhead*): ter baixo custo adicional de consumo de recursos compartilhados pelas aplicações como CPU, memória, E/S e banda de rede.

1.4.1. Ganglia

O Ganglia [MAS 2004] é um sistema hierárquico de monitoração distribuída e escalável para sistemas de alto desempenho incluindo clusters e grades. O sistemas emprega um protocolo de escuta/anúncio (*listen/announce*) baseado em comunicação em grupo

(*multicast*) para a monitoração do estado dentro dos recursos. Também utiliza uma árvore de conexões ponto-a-ponto entre nodos representantes dos clusters a fim de federar[5] os recursos e agregar os seus estados.

Dentro de cada recurso, o Ganglia usa mensagens *heartbeat* sobre um endereço de comunicação em grupo conhecido como base para o protocolo de *membership*. Um nodo continua pertencendo ao grupo pelo envio de um *heartbeat* como um sinal de que o nodo está disponível. A não recepção dentro de um intervalo de anúncios periódicos é um sinal de que o nodo não está disponível.

Ganglia federa múltiplos clusters juntos usando uma árvore de comunicação ponto-a-ponto. Nesta árvore, os nodos folha especificam um nodo em um cluster específico, enquanto os demais nodos especificam pontos de agregação. A agregação a cada ponto da árvore é feito através de um *pooling* nos nodos filhos em intervalos periódicos. Os dados de monitoração, tanto das folhas quanto do pontos de agregação, são então exportados através de uma conexão TCP ao nodo sendo requisitado seguido por uma leitura de todos os seus dados de monitoração.

Para a implementação deste sistema são utilizadas tecnologias amplamente difundidas, tais como XML para a representação dos dados. A implementação utiliza dois *daemons*. O *daemon* de monitoração (`gmond`) provê monitoração em um único cluster implementando o protocolo de escuta/anúncio e respondendo a requisições de clientes através do envio de dados de monitoração representados em formato XML. Este *daemon* executa em cada nodo do cluster. O *meta daemon* (`gmetad`), por outro lado, provê a federação do clusters.

A figura 1.6 ilustra a qualidade da apresentação dos dados finais para o usuário. Esta figura mostra a tela principal da página de relatório da grade PlanetLab [PLA 2004].

Figura 1.6: Página web mostrando dados monitorados pelo Ganglia [PLA 2004].

[5]Este termo tem o mesmo significado que integrar ou agrupar.

1.4.2. MonALISA

MonALISA (*Monitoring Agents in a Large Integrated Services Architecture*) é outro exemplo de sistema de monitoração distribuído [NEW 2003]. MonALISA é baseado na *Dynamic Distributed Services Architecture* (DDSA), que é projetada para atender requisitos de pesquisadores de área de física para a monitoração de grades.

MonALISA é um *framework* que foi implementado usando as tecnologias JINI/JAVA e WSDL/SOAP. Cada servidor MonALISA age como um serviço dinâmico e disponibiliza a funcionalidade de ser descoberto e usado por qualquer outro serviço ou cliente que precise das informações de monitoração. Este *framework* pode integrar ferramentas de monitoração e procedimentos existentes para a coleta de parâmetros descrevendo nodos computacionais, aplicações e desempenho da rede.

A figura 1.7 mostra uma das possíveis visões globais de uma grade com a representação do tráfego. Já a figura 1.8 apresenta uma árvore de nodos de grade e o menor caminho computado dinamicamente para uso em replicação de dados.

Figura 1.7: Monalisa: representação do tráfego na WAM em tempo real [MON 2004].

1.4.3. GridRM e jGMA

O GridRM [BAK 2003] é um sistema de monitoração de recursos *open-source*. o GridRM possui duas camadas: (1) Global: provê informações entre nodos da grade; e (2) Local: onde o *GridRM gateway* é o ponto de acesso para os dados. GridRM usa *GridRM gateways* para coordenar a gerência e a monitoração dos recursos em cada nodo da grade. Isto inclui o acesso controlado a dados de tempo-real e dados ao longo do tempo (histórico) obtidos sobre os recursos locais.

Os *GridRM gateways* mantém informações detalhadas sobre os produtores e o que eles disponibilizam. O GridRM usa o jGMA para distribuir informações de monitoração na grade através dos *GridRM gateways*.

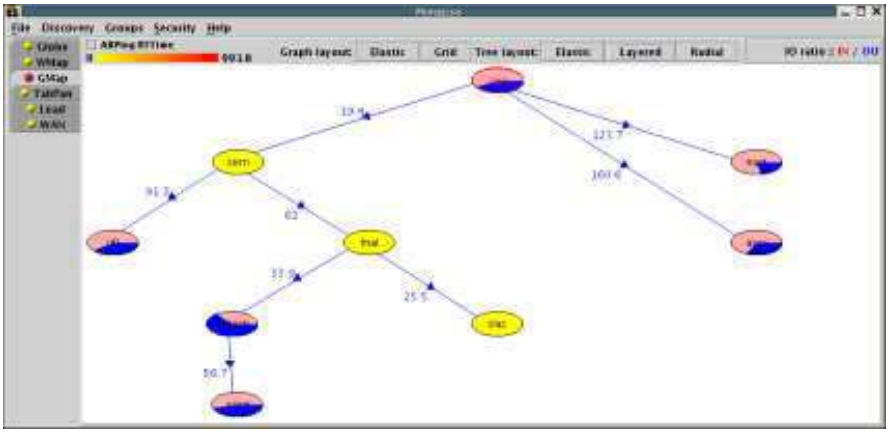

Figura 1.8: Monalisa: menor caminho computado dinamicamente [MON 2004].

jGMA [BAK 2004] é uma implementação em Java da arquitetura de monitoração proposta pelo Global Grid Forum (*Grid Monitoring Architecture* ou GMA [TIE 2002]). A GMA é uma arquitetura que busca apresentar o mínimo de especificação necessária para dar suporte as funcionalidades de monitoração e permitir a interoperabilidade entre ferramentas. Ela é baseada em um arquitetura produtor/consumidor integrada por um sistema de registro, isto é, possui três tipos de componentes: (1) serviço de diretório: suporta publicação e descoberta de informações; (2) produtor: disponibiliza dados de desempenho; e (3) consumidor: recebe dados de desempenho. A implementação do jGMA usa um componente adicional, um *servlet*, que permite comunicação remota entre os componentes. jGMA usa Apache Tomcat para comunicação entre domínios.

1.5. Gerenciamento de dados

Uma típica aplicação de grade se caracteriza por uma grande demanda de recursos, por isto antes de apresentarmos considerações sobre gerenciamento de dados, apresentamos algumas informações adicionais sobre compartilhamento de recursos.

Segundo Foster *et al* [FOS 2001], o problema real e específico que está por trás do conceito de grade é o compartilhamento de recursos, como dados ou unidades de processamento, de forma coordenada e a resolução de problemas em organizações virtuais multi-institucionais e dinâmicas.

Neste contexto, uma **organização virtual** é um conjunto de instituições e/ou indivíduos definido por regras de compartilhamento. Deste modo, é permitido que grupos distintos compartilhem recursos de uma forma controlada de tal modo que os membros podem colaborar para atingir um objetivo compartilhado.

Com relação ao compartilhamento de recursos é preciso apresentar mais alguns detalhes para permitir a compreensão dos problemas desta área. O compartilhamento é mais do que apenas troca de documentos: pode envolver o acesso remoto a programas, computadores, dados, sensores e outros recursos. Ele é condicional: cada dono de um recurso torna-o disponível sujeito a restrições sobre quando, onde e o que pode ser feito. A implementação de tais restrições requer mecanismos para expressar políticas, para estabelecer a identidade do consumidor ou recurso (autenticação) e para determinar quando uma

operação é consistente com as relações de compartilhamento aplicáveis (autorização).

Os relacionamentos de compartilhamento podem variar dinamicamente ao longo do tempo, em termos dos recursos envolvidos, da natureza do acesso permitido e dos participantes para o qual o acesso é permitido. Estas relações não precisam necessariamente envolver um conjunto nomeado explicitamente de indivíduos, mas ao invés disto podem ser definidos implicitamente por políticas que governem o acesso ao recurso. A natureza dinâmica das relações de compartilhamento significa que são exigidos mecanismos de descoberta (*discovering*) e caracterização da natureza dos compartilhamentos que existem em um determinado ponto do tempo.

Em uma perspectiva ampla, existem várias operações envolvidas para permitir o processamento e gerenciamento dos dados [STO 2001]:

- *Pré-processamento e Formatação*: O objetivo destas operações é colocar os dados não estruturados em um formato que possa ser facilmente utilizado. Metadados que correspondem a estrutura da fonte de dados (p.ex. esquema do banco de dados) são freqüentemente utilizados para facilitar a tradução de um formato para outro e para permitir a integração de múltiplas fontes de dados.

- *Fusão ou Divisão*: A fusão de dados é feita para combinar diferentes fontes de dados, provendo um conjunto de dados unificado. A divisão (*data splitting*) ocorre quando um conjunto único de dados é dividido para facilitar o processamento dos subconjuntos em paralelo.

- *Armazenamento*: Esta operação envolve o armazenamento dos dados em diferentes mídias, que variam em capacidade e "inteligência", podendo envolver migração e replicação. No contexto de grade um dos problemas é suportar a heterogeneidade de dispositivos e vendedores, além das restrições relacionadas ao desempenho que podem ou não ser levadas em conta.

- *Análise*: Pode variar da análise de dados para verificação de hipóteses à verificação da qualidade dos dados. Normalmente necessita de uma etapa anterior de pré-processamento.

- *Estimativa e Otimização de Consultas*: É possível estimar o tempo de execução de uma consulta a partir dos dados de entrada, e esta informação pode ser utilizada para decidir como executar a consulta. A otimização de consultas envolve uma série de técnicas já sedimentadas na área de banco de dados, e que são fundamentais para o bom desempenho do sistema como um todo.

- *Visualização, Navegação e Condução da Experimentação (Steering)*: Além de visualizar os resultados obtidos e navegar sobre estas informações, seria desejável permitir que os cientistas interagissem com as simulações em tempo real e dinamicamente guiassem a simulação em função de determinados parâmetros (*computational steering support*).

Assim, o gerenciamento de dados é um processo unificado formado por várias etapas, cada uma das quais com diferentes produtos e algoritmos. O suporte ao gerenciamento de dados pode então ser considerado como um conjunto de serviços que podem ser oferecidos em um sistema único ou através de um agregado de sistemas. Na próxima

seção, será apresentada uma proposta de integração de SGBDs com o ambiente de grade considerando justamente a necessidade de fornecer diferentes serviços.

Um dos pontos que precisam ser tratados para popularizar o uso de grades computacionais é um gerenciamento eficiente e transparente de dados. Inicialmente cada pesquisador desenvolveu pequenos programas para facilitar o acesso aos dados. Como uma evolução natural, agora existe uma preocupação de gerenciar bases de dados e arquivos de forma o mais transparente e eficiente possível.

Segundo Watson [WAT 2002], quase todas as aplicações de grade existentes usam arquivos e consequentemente pouco tem sido feito sobre a integração de bases de dados em ambiente de grade. Mas se o objetivo for suportar um conjunto mais amplo de aplicações, incluindo aplicações comerciais, esta integração se torna importante. Por isso nas subseções 1.5.1. e 1.5.2. são apresentados alguns aspectos de integração de bancos de dados em ambiente de grade.

1.5.1. Integração de Bancos de Dados em Ambiente de Grade

Atualmente a maior parte das aplicações de grade reportadas na literatura utiliza os dados em formatos de arquivos.

No entanto, existem outras aplicações que utilizam bancos de dados que podem ser executadas neste ambiente. Por isto, definir formas de integração de diferentes bancos de dados no ambiente de grade é fundamental. Watson [WAT 2002] apresenta uma primeira proposta de como unir na grade servidores de banco de dados distintos criando uma federação considerada como um "sistema de banco de dados virtual". A idéia central é definir como integrar os SGBDs existentes, evitando a criação de um SGBD para grade a partir do zero.

Um dos primeiros aspectos a ser considerado nesta integração é que não basta adotar ou adaptar os serviços de grade existentes para manipular arquivos. As razões são o conjunto mais rico de operações disponível nos SGBDs e a grande heterogeneidade. Com relação a heterogeneidade, ela envolve tanto a existência de diferentes paradigmas – p.ex. relacional e orientado a objetos – quanto diferentes funcionalidades fornecidas por produtos distintos dentro do mesmo paradigma – p.ex. Oracle e DB2. A definição da integração deve tentar achar um meio termo entre o suporte das funcionalidades disponibilizadas pelos diferentes produtos, buscando soluções comuns a eles. Outro nível de heterogeneidade que seria desejável tratar é o suporte a dados semi-estruturados como XML e relativamente não estruturados como artigos científicos.

O levantamento dos requisitos necessários para um SGBD para grade, apresentados por Watson [WAT 2002], consideram que não há razão para que aplicações de grade não exijam pelo menos as mesmas funcionalidade, ferramentas e propriedades de outros tipos de aplicações de banco de dados. Por isto, apresenta uma extensa lista de requisitos mínimos que inclui facilidades para consulta e atualização, transações, recuperação, replicação, restrições de integridade e versionamento. No entanto, o autor considera que outros requisitos devem ser atendidos devido a necessidade de suportar compartilhamento distribuído e em grande escala:

- escalabilidade: aplicações podem ter grande demanda por desempenho e capacidade. Por exemplo, aplicações de HEP supõem armazenamento de Petabytes de dados, em uma taxa em torno de 1 Terabyte por hora.

- tratamento de uso não previsível: possivelmente os tipos de acesso serão de difícil previsão devido ao caráter exploratório das aplicações de pesquisa.

- acesso guiado pelos metadados: este tipo de acesso prevê o acesso em duas etapas, isto é, a pesquisa com metadado é usada para localizar os bancos de dados que contêm os dados exigidos pela aplicação, e então o dado é acessado na segunda etapa.

- federação de múltiplos bancos de dados: algumas aplicações podem precisar combinar informações de múltiplos conjuntos de dados. Para atingir isto pode ser necessário o suporte para a integração de múltiplos SGBDs, por exemplo, através de consultas distribuídas.

Considerando estes requisitos, tanto as infra-estruturas de grade quanto os bancos de dados distruídos (DBMS) atuais não possuem total suporte para a integração de bancos de dados (BDs) em aplicações de grade. Por isto, está sendo proposta uma abordagem de integração através de um projeto orientado a serviços.

A figura 1.9 apresenta os diversos serviços propostos, utilizando um empacotador (*wrapper*) entre o grade e o SGBD. Esta abordagem permite aproveitar os sistemas e as bases já existentes. Obviamente, poderia existir um SGBD que fornecesse estes serviços de forma nativa. Esta lista de serviços pode não ser definitiva, uma vez que foi concebido dentro de um grupo de trabalho do Global Grid Forum (`http://www.gridforum.org`).

Figura 1.9: Interface de serviços para SGBD em grade [WAT 2002].

O ideal é que todos os serviços fossem disponibilizados e adotassem uma forma padronizada, mas nem sempre isto pode ser possível. Por isto, cada SGBD deve disponibilizar um serviço de metadados que forneça informações sobre a gama de serviços e operações que ele suporta.

Uma vez que a interface de serviços propostas esteja disponível em cada um dos SGBDs, a criação da federação de SGBDs ou do "sistema de banco de dados virtual" consiste na criação de um SGBD virtual que não armazena dados, mas que possui a mesma interface de serviços. Uma invocação a este SGBD virtual é tratada por uma camada intermediária (suporte aos serviços federados ou *Service Federation Middleware*) que interage com as interfaces de serviço dos SGBDs, conforme ilustrado na figura 1.10. A complexidade deste *middleware* varia conforme o serviço e em geral aumenta de forma proporcional ao aumento de heterogeneidade dos SGBDs.

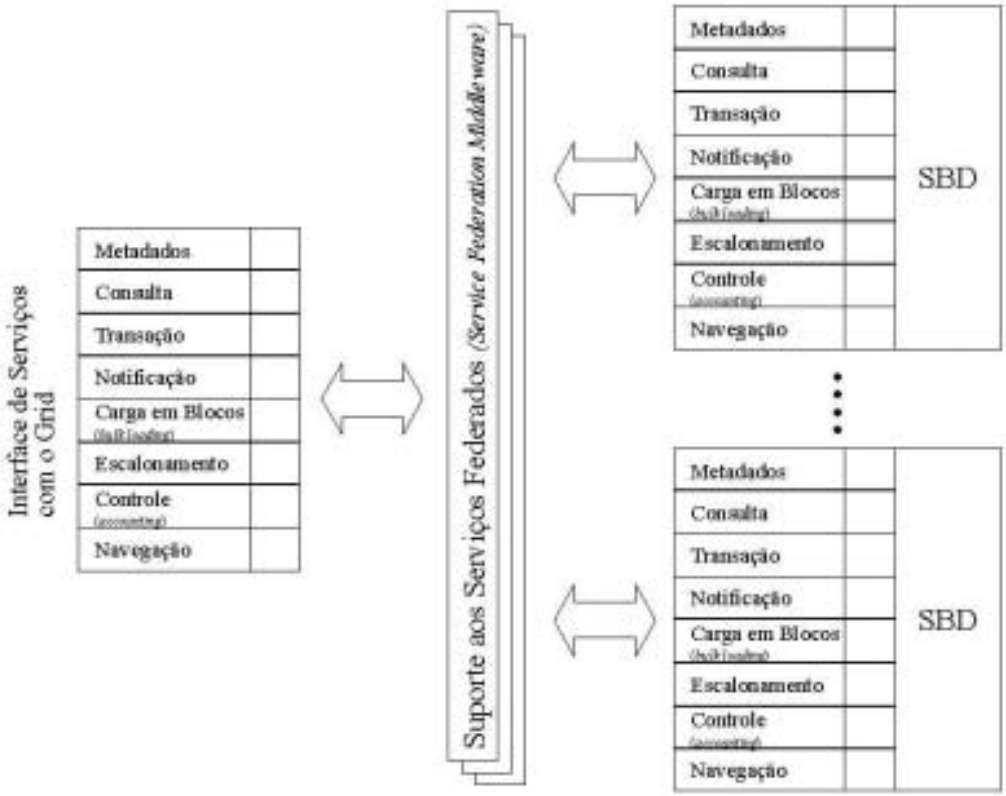

Figura 1.10: Proposta de sistema de banco de dados virtual [WAT 2002].

1.5.2. Gerenciamento de Dados em grade versus DBMS

Através dos vários trabalhos existentes na literatura e principalmente do artigo de Stockinger [STO 2001a], é possível traçar algumas diferenças entre bancos de dados distribuídos (DBMS) e as demandas em ambiente de grade.

Considerando inicialmente a replicação de dados, temos a sincronização das réplicas em DBMS baseado em transações pequenas formadas por operações de leitura e/ou escrita. Por outro lado, em aplicações como HEP temos aplicações que produzem dados gerando um arquivo inteiro no que pode ser considerado como uma única transação longa.

Em DBMS, as funções de custo para definir a replicação não consideram a carga do servidor ou da rede, bem como a quantidade de dados envolvida é relativamente pequena. Logo, em grade seria preciso repensar as funções de custo utilizadas na definição da replicação.

Em DBMS, normalmente há uma mesma forma de acessar dados, seja para escrita ou para leitura. No caso do ambiente de grade esta pode não ser uma boa alternativa dependendo do número de objetos a ser acessado. Para acessar grandes quantidades de dados, o ideal seria utilizar um mecanismo de transferência de arquivos para permitir o acesso local aumentando a eficiência em termos de tempo de resposta.

Com relação aos serviços de diretório e catálogo para acesso a réplicas, temos a necessidade de metadados para fornecer informações sobre a distribuição e acesso aos dados. DBMSs possuem métodos internos de acesso a estes metadados, enquanto no ambiente de grade estas estruturas de dados são fornecidas explicitamente por serviços disponibilizados ao usuário.

Existem vários trabalhos que afirmam que as bases de dados em grade podem chegar a serem usadas em 90% dos casos apenas para leitura, facilitando, por exemplo, a replicação. No entanto, é preciso não esquecer que existirão casos de alteração e portanto técnicas de sincronização e consistência utilizadas em DBMS precisam ser adaptadas para garantir a correção dos resultados.

Com relação ao padrão de acesso aos dados, em um DBMS é possível obter estatísticas e prever os possíveis comportamentos das aplicações realizando otimizações baseadas nestas informações. No caso do um ambiente de grade isto pode ou não se aplicar, uma vez que o caráter de pesquisa das atuais aplicações pressupõe experimentações bastante variadas com padrões de acesso não previsto para as consultas.

Finalmente, a otimização de consultas em ambientes distribuídos que trabalham em ambientes de rede local, podem considerar os custos de comunicação como uma parcela não tão significativa do custo total de processamento da consulta (Özsu e Valduriez, p.193 [ÖZS 99]). No ambiente de grade, no entanto, o custo de comunicação continua sendo um fator determinante do custo total de obtenção e manipulação dos dados e, consequentemente, do custo de execução da consulta.

1.6. Considerações Finais

A principal motivação para o estudo de grades advém das grandes aplicações de áreas científicas que não podem ser satisfeitas com a tecnologia de hardware atualmente disponível.

Para a realidade brasileira, e mais especificamente regional, há ainda uma forte motivação econômica. Quase todas as universidades que realizam pesquisas em processamento paralelo e distribuído conseguem manter conjuntos de máquinas em redes locais ou clusters que são suficientes para pequenos experimentos. Computações mais complexas exigiriam um conjunto mais poderoso de máquinas que dificilmente estariam disponíveis localmente mas que poderiam ser obtidos pelo agrupamento de recursos de diferentes instituições. Na prática, quase todas as universidades da região sul que realizam pesquisas em processamento paralelo e distribuído mantém algum tipo de colaboração, formal ou informal, o que facilita a criação de grades. Uma infra-estrutura de grade permitiria a integração destes recursos dispersos geograficamente viabilizando as atividade de

pesquisa mais complexas.

É importante notar que, embora a primeira vista uma grade possa parecer apenas um grande aglomerado de computadores, a proposta de computação em grade pressupõe uma grande demanda por ciclos computacionais e/ou capacidade de armazenamento e uma visão de gerenciamento muito mais complexa.

Este texto apresentou apenas alguns conceitos e trabalhos existentes na área. A partir deste texto e das indicações constantes nas referências bibliográficas acredita-se ser possível ter um bom ponto de partida para estudos nesta área.

Agradecimentos

A pesquisa que originou este texto foi parcialmente financiada pelo CNPq.

1.7. Bibliografia

[ALL 2002] ALLCOCK, W. et al. **GridFTP protocol specification**. Technical Report. Available at `http://www.globus.org/research/papers/GridftpSpec02.doc`.

[ALL 2002a] ALLCOCK, W. et al. **GridFTP update january 2002**. Technical Report. Available at `http://www.globus.org/datagrid/deliverables/GridFTP-Overview-200201.pdf%`.

[AND 2003] ANDRADE, N. et al. OurGrid: an approach to easily assemble grids with equitable resource sharing. In: WORKSHOP ON JOB SCHEDULING STRATEGIES FOR PARALLEL PROCESSING, 9., 2003. **Proceedings...** [S.l.: s.n.], 2003.

[AVA 2004] AVAKI. `http://www.avaki.com/`.

[BAK 2000] BAKER, M.; BUYYA, R.; LAFORENZA, D. The Grid: international efforts in global computing. In: INTERNATIONAL CONFERENCE ON ADVANCES IN INFRASTRUCTURE FOR ELECTRONIC BUSINESS, SCIENCE, AND EDUCATION ON THE INTERNET (SSGRR 2000), 2000, Rome, Italy. **Anais...** [S.l.: s.n.], 2000.

[BAK 2004] BAKER, M.; GROVE, M. jgma: a lightweight implementation of the grid monitoring architecture. In: UKUUG LISA/WINTER CONFERENCE, 2004, Southampton, UK. **Anais...** [S.l.: s.n.], 2004.

[BAK 2003] BAKER, M.; SMITH, G. Gridrm: an extensible resource management system. In: IEEE INTERNATIONAL CONFERENCE ON CLUSTER COMPUTING (CLUSTER 2003), 2003, Hong Kong, China. **Anais...** [S.l.: s.n.], 2003.

[BAR 2002] BARBOSA, J. L. V. et al. Holoparadigm: a multiparadigm model oriented to development of distributed systems. In: INTERNATIONAL CONFERENCE ON PARALLEL AND DISTRIBUTED SYSTEMS (IC-

PADS 2002), 9., 2002, Taiwan, ROC. **Proceedings...** IEEE Press, 2002. p.165–170.

[BER 2003] BERMAN, F.; FOX, G.; HEY, T. **Grid computing**: making the global infrastructure a reality. 1.ed. [S.l.]: John Wiley & Sons Inc., 2003. 1060p.

[CAV 2004] Cave5D release 1.4. `http://www.ccpo.odu.edu/~cave5d/`.

[CHA 99] CHAPIN, S. J. et al. Resource management in Legion. In: WORKSHOP ON JOB SCHEDULING STRATEGIES FOR PARALLEL PROCESSING (JSSPP '99), IN CONJUNCTION WITH IPDPS '99, 5., 1999. **Proceedings...** [S.l.: s.n.], 1999.

[CIR 2001] CIRNE, W.; MARZULLO, K. OpenGrid: a user-centric approach for grid computing. In: SYMPOSIUM ON COMPUTER ARCHITECTURE AND HIGH PERFORMANCE COMPUTING (SBAC-PAD 2001), 13., 2001. **Proceedings...** [S.l.: s.n.], 2001.

[MON 2004] CLIENT examples – real-time monitoring of distributed farm. `http://monalisa.cacr.caltech.edu/demos_client_examples.html`.

[CZA 98] CZAJKOWSKI, K. et al. A resource management architecture for metacomputing systems. In: IPPS/SPDP '98 WORKSHOP ON JOB SCHEDULING STRATEGIES FOR PARALLEL PROCESSING, 1998. **Proceedings...** [S.l.: s.n.], 1998. p.62–82.

[DEF 96] DEFANTI, T. A. et al. 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.

[DIS 2004] DISTRIBUTED.NET. `http://www.distributed.net/`.

[FAF 2000a] FAFNER overview. `http://www.npac.syr.edu/factoring/overview.html`.

[FOS 2002] FOSTER, I. et al. **The physiology of the Grid**: an open grid services architecture for distributed systems integration. Available at `http://www.globus.org/research/papers/ogsa.pdf`.

[FOS 2001] FOSTER, I.; KESSELMAN, C.; TUECKE, S. 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.

[FOS 98] FOSTER, I.; KESSELMAN, C. **The Grid**: blueprint for a new computing infrastructure. 1.ed. San Francisco, California, USA: Morgan Kaufmann, 1998. 701p.

[FOS 98a] FOSTER, I.; KESSELMAN, C. Computational Grids. In: FOSTER, I.; KESSELMAN, C. (Eds.). **The Grid**: blueprint for a new computing infrastructure. 1.ed. San Francisco, California, USA: Morgan Kaufmann, 1998. p.15–52.

[FOS 98b] FOSTER, I.; KESSELMAN, C. The Globus project: a status report. In: IPPS/SPDP '98 HETEROGENEOUS COMPUTING WORKSHOP, 1998. **Proceedings...** [S.l.: s.n.], 1998. p.4–18.

[FRE 2002] FREY, J. et al. Condor-G: a computation management agent for multi-institutional grids. **Cluster Computing**, v.5, p.237–246, 2002.

[GRI 99] GRIMSHAW, A. S. et al. Wide-area computing: resource sharing on a large scale. **IEEE Computer**, v.32, n.5, p.29–36, May 1999.

[KRA 2002] KRAUTER, K.; BUYYA, R.; MAHESWARAN, M. A taxonomy and survey of grid resource management systems for distributed computing. **Software: Practice and Experience**, v.32, n.2, p.135–164, February 2002.

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

[MAS 2004] MASSIE, M. L.; CHUN, B. N.; CULLER, D. E. The ganglia distributed monitoring system: design, implementation, and experience. **Parallel Computing**, v.30, n.7, p.817–840, July 2004.

[NEM 2002] NEMETH, Z.; SUNDERAM, V. A formal framework for defining grid systems. In: SECOND IEEE/ACM INTERNATIONAL SYMPOSIUM ON CLUSTER COMPUTING AND THE GRID (CCGRID2002), 2002, Berlin, Germany. **Proceedings...** [S.l.: s.n.], 2002. p.202–211.

[NEW 2003] NEWMAN, H. B. et al. MonALISA: a distributed monitoring service architecture. In: COMPUTING IN HIGH ENERGY AND NUCLEAR PHYSICS (CHEP03), 2003, La Jolla, California, USA. **Anais...** [S.l.: s.n.], 2003.

[PAR 2003] PARANHOS, D. S.; CIRNE, W.; BRASILEIRO, F. V. Trading cycles for information: using replication to schedule bag-of-tasks applications on computational grids. In: INTERNATIONAL CONFERENCE ON PARALLEL AND DISTRIBUTED COMPUTING (EURO-PAR 2003), 26., 2003. **Proceedings...** [S.l.: s.n.], 2003.

[PLA 2004] PLANET lab grid report. `http://planetlab.millennium.berkeley.edu/ganglia_v2/`.

[REA 2003] REAL, R. A. et al. Resource scheduling on grid: handling uncertainty. In: IEEE/ACM 4TH INTERNATIONAL WORKSHOP ON GRID COMPUTING, 2003, Phoenix, Arizona. **Proceedings...** [S.l.: s.n.], 2003.

[REA 2003a] REAL, R. A. et al. Tratamento da incerteza no escalonamento de recursos em pervasive computing. In: CONFERÊNCIA IADIS IBERO-AMERICANA WWW/INTERNET, 2003, Algarve, Portugal. **Anais...** [S.l.: s.n.], 2003. p.167–170.

[ROU 2003] ROURE, D. D. et al. The evolution of the Grid. In: BERMAN, F.; FOX, G.; HEY, T. (Eds.). **Grid computing**: making the global infrastructure a reality. [S.l.]: Wiley & Sons, 2003. p.65–100.

[FAF 2000] RSA130: getting started with FAFNER. `http://cs-www.bu.edu/cgi-bin/FAFNER/factor.pl`.

[SAN 2003] SANDHOLM, T.; GAWOR, J. **Globus Toolkit 3 Core – A Grid Service Container Framework**. White paper. Available at `http://www-unix.globus.org/toolkit/3.0beta/ogsa/docs/gt3_core.pdf`.

[SET 2004] SETI@HOME – the search for extraterrestrial intelligence at home. `http://setiathome.ssl.berkeley.edu/`.

[STO 2001] STOCKINGER, H. et al. Data management for grid environments. In: EUROPEAN HIGH PERFORMANCE COMPUTING & NETWORKING (HPCN 2001), 2001, Amsterdam, The Netherlands. **Anais...** [S.l.: s.n.], 2001.

[STO 2001a] STOCKINGER, H. Distributed database management systems and the data grid. In: EIGHTEENTH IEEE SYMPOSIUM ON MASS STORAGE SYSTEMS, 2001, Hyatt Regency Islandia, San Diego, USA. **Anais...** [S.l.: s.n.], 2001.

[TAN 95] TANENBAUM, A. S. **Distributed operating systems**. [S.l.]: Prentice Hall, 1995.

[TAN 2002] TANNENBAUM, T. et al. Condor – a distributed job scheduler. In: STERLING, T. (Ed.). **Beowulf cluster computing with Linux**. [S.l.]: MIT Press, 2002.

[THA 2003] THAIN, D.; TANNENBAUM, T.; LIVNY, M. Condor and the Grid. In: BERMAN, F.; FOX, G.; HEY, T. (Eds.). **Grid computing**: making the global infrastructure a reality. [S.l.]: John Wiley, 2003.

[The 2000] The Globus Project. **GridFTP**: universal data transfer for the grid. Technical Report. Available at `http://www.globus.org/datagrid/deliverables/C2WPdraft3.pdf`.

[WSR 2004] The WS-Resource Framework. `http://www.globus.org/wsrf/`.

[TIE 2002] TIERNEY, B. et al. **A grid monitoring architecture**. 13p. Available at `http://www-didc.lbl.gov/GGF-PERF/GMA-WG/papers/GWD-GP-16-3.pdf`.

[TUE 2002] TUECKE, S. et al. **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`.

[WAT 2002] WATSON, P. **Databases and the grid**. [S.l.]: National e-Science Centre, 2002. UK e-Science Programme Technical Report Series, Document produced by "Database Access and Integration Services Working Group"of Global Grid Forum. Available at `http://www.cs.man.ac.uk/grid-db/papers/dbg.pdf`. (UKeS-2002-01).

[WEL 2003] WELCH, V. et al. Security for grid services. In: IEEE INTERNATIONAL SYMPOSIUM ON HIGH PERFORMANCE DISTRIBUTED COMPUTING (HPDC'03), 12., 2003, Seattle, Washington. **Anais...** [S.l.: s.n.], 2003. Also available at `http://www.globus.org/security/GSI3/GT3-Security-HPDC.pdf`.

[WRI 2001] WRIGHT, D. Cheap cycles from the desktop to the dedicated cluster: combining opportunistic and dedicated scheduling with condor. In: CONFERENCE ON LINUX CLUSTERS: THE HPC REVOLUTION, 2001, Champaign - Urbana, IL, USA. **Proceedings...** [S.l.: s.n.], 2001.

[YAM 2002] YAMIN, A. et al. ISAM: a pervasive view in distributed mobile computing. In: IFIP TC6 / WG6.2 & WG6.7 CONFERENCE ON NETWORK CONTROL AND ENGINEERING FOR QOS, SECURITY AND MOBILITY (NET-CON 2002), 2002. **Proceedings...** [S.l.: s.n.], 2002. p.431–436.

[YAM 2003] YAMIN, A. et al. Towards merging context-aware, mobile and grid computing. **International Journal of High Performance Computing Applications**, London, v.17, n.2, p.191–203, June 2003.

[YAM 2002] YAMIN *et al*, A. A framework for exploiting adaptation in high heterogeneous distributed processing. In: XIV SYMPOSIUM ON COMPUTER ARCHITECTURE AND HIGH PERFORMANCE COMPUTING (SBAC-PAD), 2002, Vitória, ES. **Proceedings...** [S.l.: s.n.], 2002.

[ÖZS 99] ÖZSU, M. T.; VALDURIEZ, P. **Principles of distributed database systems**. 2.ed. [S.l.]: Prentice Hall, 1999.