Glauco Antonio Ludwig

Uma Abordagem baseada em Políticas para Contabilização e Caracterização de Uso Global de Grades Computacionais

Glauco Antonio Ludwig

Uma Abordagem baseada em Políticas para Contabilização e Caracterização de Uso Global de Grades Computacionais

Dissertação apresentada à Universidade do Vale do Rio dos Sinos (UNISINOS) como requisito parcial para a obtenção do título de Mestre em Computação Aplicada

Orientador: Prof. Dr. Luciano Paschoal Gaspary

São Leopoldo

2006

Ficha catalográfica elaborada pela Biblioteca da

Universidade do Vale do Rio dos Sinos

L948a Ludwig, Glauco Antonio

Uma abordagem baseada em políticas para contabilização e caracterização de uso global de grades computacionais / por Glauco Antonio Ludwig. – 2006.

74 f. : il. ; 30 cm.

Dissertação (mestrado) — Universidade do Vale do Rio dos Sinos, Programa de Pós-Graduação em Computação Aplicada, 2006.

"Orientação: Prof. Dr. Luciano Paschoal Gaspary, Ciências Exatas e Tecnológicas".

1. Sistemas distribuídos. 2. Grades computacionais. I. Título.

CDU 004.75

Catalogação na Publicação:

Bibliotecária Eliete Mari Doncato Brasil - CRB 10/1184

Espaço reservado para folha de aprovação impressa.

“I’ve been told that to get what you want

You just gotta give what you can.”

Dexter Holland

Agradecimentos

Primeiramente, quero agradecer a toda minha família: Mãe (Síria), Pai (Romeu), Mana (Isa), Nando (sobrinho) e Russo (cunhado), e a minha namorada Katiele. Sem o imenso e incansável apoio de vocês tudo seria mais difícil. Eu amo vocês.

Agradeço ao meu orientador Prof. Dr. Luciano Paschoal Gaspary, por ter acreditado em mim e por toda a sua dedicação ao longo da elaboração deste trabalho. Muito obrigado por todos os seus conselhos e ensinamentos.

Da mesma forma, agradeço ao Prof. Dr. Gerson Geraldo Homrich Cavalheiro pelas suas contribuições para minha formação e para esta pesquisa.

Sou grato também a Hewlett-Packard Brasil pela bolsa integral que viabilizou a realização do mestrado com dedicação exclusiva. Valeu muito pela oportunidade oferecida.

Enfim, agradeço a todos os meus amigos e amigas que, de uma forma ou de outra, estiveram ao meu lado em mais este desafio. Agradeço aos colegas de pesquisa e amigos Sidnei Franco, Juliano Freitas, Ricardo Sanchez e Débora Alves pela troca de idéias e ajuda prestada. Fica um registro especial aos colegas e amigos do mestrado: Leonardo Fagundes, Marcelo Scopel e Gilberto Muller.

Resumo

Contabilizar e caracterizar o uso de grades computacionais constitui uma tarefa de gerenciamento indispensável, especialmente quando essas grades são usadas em larga escala, envolvendo várias instituições e participantes. As soluções de gerência de grades existentes atualmente são limitadas: (a) por se destinarem a monitorar apenas o estado dos recursos que compõem o ambiente (como consumo de memória e espaço de armazenamento em disco), omitindo dados estatísticos e históricos sobre a execução de aplicações; (b) por não oferecem suporte à coleta, ao processamento e à consolidação de informações que são geradas por diferentes tecnologias de grade; e (c) por não oferecem um esquema seletivo para a divulgação das informações gerenciais que são obtidas. Para suprir as lacunas recém mencionadas, esta dissertação propõe uma abordagem baseada em políticas para contabilização e caracterização de uso de grades computacionais compostas por sistemas heterogêneos. A abordagem proposta se materializa através de uma arquitetura baseada no padrão *Web Services Distributed Management* (WSDM). A dissertação apresenta essa arquitetura, bem como resultados obtidos com a sua instanciação em um ambiente real.

Palavras-chave: grades computacionais, contabilização, caracterização, aplicações.

Abstract

Accounting and characterizing the use of computational grids constitutes a management task of paramount importance, especially when they are used in large scale, involving many institutions and participants. Current grid computing management solutions are limited since they: (a) solely monitor the status of environment resources (like CPU and disk space consumption), omitting statistical and historical data about the execution of applications; (b) do not offer support for gathering, processing and consolidation of information that is generated by heterogeneous grid technologies; and (c) do not allow the specification of policies to distribute the collected information. To fullfil this gap, this work proposes an approach based on polices to account and characterize usage of grid computing infrastructures, even when such grids are formed by heterogeneous middleware. The approach is materialized through an architecture based on the Web Services Distributed Management standard. This work presents the architecture proposed and results obtained with its instantiation in a real setup.

Key-words: computational grids, accounting, characterization, applications.

Lista de Figuras

Figura 2.1. Relação entre GT4, OGSA, WSRF e serviços *web*.....20
Figura 2.2. Comparação entre um serviço *web* tradicional e uma aplicação Globus.....21
Figura 2.3. Arquitetura de um Condor *pool* com seus *daemons*. 24
Figura 2.4. Componentes da infra-estrutura OurGrid. 26
Figura 2.5. Gerenciamento de recursos através de MUWS.....29
Figura 2.6. WS-Notification sem o uso de um *NotificationBroker*.....30
Figura 2.7. WS-Notification fazendo uso de um *NotificationBroker*.....31
Figura 3.1. Arquitetura de VisPerf e a interação entre seus componentes.....35
Figura 3.2. Componenentes da arquitetura GMA e suas interações.....36
Figura 3.3. Arquitetura GridRM e a interação entre seus componentes. 37
Figura 3.4. Arquitetura Remos e a interação entre seus componentes.....38
Figura 3.5. Interação entre componentes da arquitetura de MonLISA. 39
Figura 3.6. Interação entre os componentes da arquitetura de Ganglia.....40
Figura 4.1. Grade formada por instituições que fazem uso de diferentes tecnologias. 46
Figura 4.2. Visão geral da arquitetura e sua instanciação em ambiente de grade. 47
Figura 4.3. Formato do conteúdo de uma requisição SETRESOURCEPROPERTIES. 52
Figura 4.4. Componenetes de uma política de divulgação de informações. 52
Figura 4.5. Formato simplificado de uma subscrição para a propriedade JOBFAILED. 54
Figura 4.6. Formato de uma notificação recebida pela aplicação de gerenciamento. 54
Figura 4.7. Formato do retorno de uma requisição do tipo GETRESOURCEPROPERTY.....55
Figura 4.8. Estrutura dos *logs* MyGrid e informações de interesse para o *publisher*.....57
Figura 4.9. Requisições SETRESOURCEPROPERTIES geradas a partir de eventos capturados de *logs* OurGrid. 57
Figura 4.10. Interface gráfica da aplicação de gerenciamento. 59
Figura 4.11. Pop-up para realizar subscrições.....60
Figura 5.1. Infra-estrutura de grade utilizada para a avaliação da arquitetura. 62
Figura 5.2. Número de *jobs* executados x *Institution B* ExecHost.....65
Figura 5.3. Número de *jobs* executados ao longo do dia 23 de novembro de 2005.....65

Lista de Tabelas

Tabela 3.1. O modelo de informações Usage Record. 41
Tabela 3.2. Comparação entre os trabalhos relacionados a esta proposta. 42
Tabela 4.1. Propriedades fornecidas por GUACS. 51
Tabela 4.2. Exemplo de políticas definidas no serviço de autorização. 53
Tabela 5.1. Políticas definidas no serviço de autorização de *Institution A* e *Institution B.* 63
Tabela 5.2. Políticas definidas no serviço de autorização de *Institution C.* 63

Lista de Abreviaturas e Siglas

BoT	Bag of Task
CPU	Central Processing Unit
DMTF	Distributed Management Task Force
FTP	File Transfer Protocol
GGF	Global Grid Forum
GMA	Grid Monitoring Architecture
GRAM	Globus Resource Allocation Manager
GridRM	Grid Resource Monitoring
GSI	Grid Security Infrastructure
GT	Globus Toolkit
GuM	Grid Machine
GuMP	Grid Machine Provider
MonALISA	Monitoring Agents in a Large Integrated Services Architecture
MOWS	Managing of Web Services
MUWS	Managing Using Web Services
NWS	Network Weather Service
OGSA	Open Grid Services Architecture
RBAC	Role-based Access Control
REMOS	Resource Monitoring System
RMS	Resource Management System
RUS	Resource Usage Service
SOAP	Simple Object Access Protocol
SSH	Security Shell
UR	Usage Record
XACML	eXtensible Access Control Markup Language
XML	Extensible Markup Language
WS	Web Service
WSDL	Web Services Description Language

WSDM	Web Services Distributed Management
WSMF	Web Services Management Framework
WSRF	Web Services Resource Framework

Sumário

1 Introdução **13**
1.1 Contextualização *13*
1.2 Definição do problema *14*
1.3 Objetivos *15*
1.4 Organização da dissertação *16*
2 Gerenciamento de Grades Computacionais: Conceitos e Tecnologias **17**
2.1 Em busca de um conceito para computação em grade *17*
2.2 Sistemas para computação em grade *18*
2.2.1 Globus 18
2.2.2 Condor 22
2.2.3 OurGrid 24
2.3 Benefícios e requisitos de gerenciamento em ambientes de grade *26*
2.4 Frameworks de gerenciamento baseados em serviços web *27*
2.4.1 Web Services Distributed Management (WSDM) 28
2.4.2 Web Services for Management (WS-Management) 31
2.4.3 Considerações sobre WSDM e WS-Management 32
2.5 Sumário *33*
3 Trabalhos relacionados **34**
3.1 visPerf *34*
3.2 Grid Monitoring Architecture (GMA) *35*
3.3 Grid Resource Monitoring (GridRM) *36*
3.4 Resource Monitoring System (Remos) *37*
3.5 MonALISA *38*
3.6 Ganglia *39*
3.7 Usage Record (UR) e Resource Usage Service (RUS) *40*
3.8 Considerações gerais *42*
4 Grid Usage Accounting and Characterization Service **44**
4.1 Requisitos *44*

4.2 Visão conceitual *46*
4.3 Componentes *49*
4.3.1 Serviço de Contabilização e Caracterização 49
4.3.2 Publishers 51
4.3.3 Serviço de Autorização 52
4.3.4 Aplicação de Gerenciamento 53
4.4 Implementação *55*
4.4.1 Serviço de Contabilização e Caracterização 55
4.4.2 Publishers 56
4.4.3 Serviço de Autorização 58
4.4.4 Aplicação de Gerenciamento 58
5 Avaliação Experimental 61
5.1 Instanciação da arquitetura *61*
5.2 Resultados obtidos *64*
6 Considerações Finais 67
Referências Bibliográficas 69

1 Introdução

1.1 Contextualização

O interesse no uso da tecnologia de grades computacionais vem se expandido de forma gradual e consistente nos últimos anos. Projetadas para o compartilhamento de recursos distribuídos, o modelo usado pelas infra-estruturas de grades apresenta algumas motivações: (a) reduz a necessidade de atualização de equipamentos, fazendo uso de computadores geograficamente dispersos; (b) fornece ganho de desempenho na solução de problemas que exigem processamento intensivo; (c) reduz o uso impróprio de recursos, tirando melhor proveito, por exemplo, dos ciclos ociosos de processamento dos computadores [Foster, 1998]; (d) provê transparência no uso dos recursos, considerando-os como um único e poderoso computador [Nemeth, 2002]; (e) oferece a disponibilidade de acesso apenas a recursos específicos de um nodo, ou seja, acesso a um nodo não significa acesso a todos os recursos do nodo [Nemeth, 2002]. Observa-se, assim, que a tecnologia de grade está fortemente relacionada a de sistemas distribuídos e, também, de aglomerados de computadores (*clusters*), uma vez que essas três tecnologias apresentam características em comum. Contudo, ela se distingue de sistemas distribuídos nos itens (c) e (d) e, de *clusters*, nos itens (a) e (c).

O surgimento de aplicações que exigem cada vez mais um elevado poder computacional, tais como as de alinhamento de seqüências de genomas e simulações de dinâmica molecular, somado aos benefícios trazidos pela computação em grade, levaram essa tecnologia a adquirir crescente importância. Com o amadurecimento das soluções hoje oferecidas, a expectativa é que nos próximos três a cinco anos essa tecnologia passe a ser intensamente aproveitada não somente no ambiente acadêmico, mas sobretudo no corporativo [Berman, 2005]. Nessa direção, um aspecto chave para a consolidação e a franca utilização de grades computacionais (especialmente em ambientes corporativos) é a disponibilização de soluções que permitam gerenciar a infra-estrutura implantada.

Quando usadas em larga escala, envolvendo várias instituições e participantes, torna-se importante – além de realizar gerenciamento de falhas, configuração, desempenho e segurança – contabilizar e caracterizar o uso dessa infra-estrutura para:

- obter informações detalhadas sobre as aplicações executadas (i.e. origem, onde foram executadas, tempo de execução, recursos que consumiram e usuários que as executaram);
- verificar o perfil dos usuários, analisando quais contribuem com mais recursos e quais os que fazem mais uso do poder computacional (possibilitando cobrá-los pelo acesso aos recursos da grade se este for o caso);
- identificar a execução de aplicações maliciosas (uma aplicação executando por um tempo extremamente prolongado poderia indicar a intenção de apenas consumir recursos da grade, impedindo seu uso legítimo);
- garantir o escalonamento de aplicações e a alocação de recursos conforme o histórico dos usuários e seus créditos de utilização (usuários que tenham excedido uma cota de uso não deveriam ter acesso aos recursos da grade);
- identificar estações que não estão contribuindo de forma produtiva (uma máquina recebe tarefas para computar e acaba falhando o processamento dessas freqüentemente);
- acompanhar a evolução do seu uso (permitindo reconhecer padrões e tendências).

1.2 Definição do problema

Atualmente, existem várias soluções [Dinda, 2001; Tierney, 2002; Baker, 2003; Lee, 2003; Newman, 2003; Massie, 2004] que, de alguma forma, oferecem informações sobre o uso de grades computacionais. No entanto, a maioria delas está limitada a monitorar o estado dos recursos disponíveis no ambiente (como carga de CPU e consumo de memória), omitindo dados estatísticos e históricos sobre a execução de aplicações. Além disso, essas soluções não permitem relacionar o consumo dos recursos com as aplicações executadas sobre os mesmos. Por exemplo, com as ferramentas atuais é possível visualizar que uma determinada estação esteve com sua carga de CPU extremamente elevada nas últimas doze horas. Contudo, não é possível ter um conhecimento preciso da razão pela qual isso está ocorrendo, ou seja, há uma desvinculação entre os recursos e as aplicações que foram computadas.

Outro problema observado é que diferentes tecnologias de grade, tais como Globus [Globus, 2005], Condor [Condor, 2005] e OurGrid [OurGrid, 2005], empregam indicadores próprios para reportar a execução de aplicações (por exemplo, em um determinado sistema

uma aplicação é identificada por “idJob” e em outro por “JobId”). Assim, instituições[1] que cooperam utilizando *middlewares* de grade distintos – situação comum atualmente – acabam encontrando dificuldades para trocar informações entre si e, deste modo, obter uma visão global e uniforme de uso da infra-estrutura. Nesse sentido, as soluções de gerenciamento de grades existentes pecam por não oferecem suporte à coleta, ao processamento e à consolidação de informações que são geradas por diferentes tecnologias.

Outra limitação significativa das soluções existentes reside na ausência de suporte a um esquema de divulgação seletiva das informações coletadas. Ao mesmo tempo em que as instituições envolvidas em uma grade têm interesse em compartilhar informações a fim de obter uma visão de uso integrada da infra-estrutura, cada uma delas pode estar sendo regida por diferentes políticas de segurança. Nesse contexto, a disseminação das informações obtidas por um sistema de contabilização e caracterização pode conflitar com as políticas de segurança empregadas em uma determinada instituição. Para superar essa limitação, uma solução deve permitir que cada instituição determine, autonomamente, políticas que especifiquem as informações a serem compartilhadas e com quem.

1.3 Objetivos

Para suprir as deficiências enumeradas na seção anterior, este trabalho teve por objetivo especificar, implementar e avaliar uma arquitetura para a contabilização e a caracterização do uso global de grades computacionais.

A arquitetura foi projetada para suportar a contabilização e a caracterização do uso em ambientes de grade compostos por diferentes sistemas, tais como Globus [Globus, 2005], Condor [Condor, 2005] e OurGrid [OurGrid, 2005]. Outra característica apresentada é o suporte para a coleta de informações sobre a execução de aplicações e o consumo de recursos, de forma que seja possível associar essas duas classes de informações entre si. Uma terceira característica da arquitetura é a disponibilização de um mecanismo que permite a definição de políticas de divulgação de informações por parte das instituições envolvidas, a fim de que as mesmas especifiquem quem pode ter acesso e a quais informações.

Em consonância com as tecnologias atuais de grades – que têm procurado organizar seus componentes na forma de serviços *web* – a arquitetura foi desenvolvida com base no

---

[1] A palavra *instituição* será usada ao longo da dissertação com um significado mais amplo, podendo denotar uma organização ou, simplesmente, um departamento ou setor.

padrão WSDM (*Web Services Distributed Management*) [Vambenepe et al. 2005; Sedukhin et al. 2005], padronizado pelo OASIS[2] em março de 2005. É importante destacar que, até onde se sabe, este é o primeiro trabalho a propor um serviço para gerenciamento de grades através do *framework* WSDM.

1.4 Organização da dissertação

O restante do trabalho está organizado conforme listado a seguir. O Capítulo 2 apresenta conceitos básicos que são relevantes no decorrer da dissertação. O Capítulo 3 introduz trabalhos relacionados, que descrevem algumas soluções de gerenciamento para grades computacionais. O Capítulo 4 apresenta a arquitetura proposta, abrangendo a motivação, a arquitetura conceitual e o protótipo desenvolvido. A avaliação experimental do protótipo é descrita no Capítulo 5. A dissertação é encerrada no Capítulo 6 com considerações finais e perspectivas de trabalhos futuros.

[2] OASIS é um comitê global que dirige o desenvolvimento, a convergência e a adoção de padrões para *e-business*.

2 Gerenciamento de Grades Computacionais: Conceitos e Tecnologias

Este capítulo apresenta em sua Seção 2.1 um embasamento teórico sobre grades computacionais e, em sua Seção 2.2, alguns dos principais sistemas que dão suporte a esse tipo de computação. Na Seção 2.3 são relatados os principais requisitos e necessidades que levam à busca por soluções para o gerenciamento dessas grades. Por fim, a Seção 2.4 apresenta duas propostas que têm como objetivo oferecer suporte para o gerenciamento de recursos de forma padronizada, através da tecnologia de serviços *web*.

2.1 Em busca de um conceito para computação em grade

A idéia de usufruir sob demanda da capacidade de computação geograficamente dispersa, sem a preocupação de qual a sua origem e de como é mantida, acabou impulsionando o conceito de computação em grade. De acordo com Foster [Foster et al., 2001], as grades foram pensadas na metade da década de 1990 como uma infra-estrutura computacional distribuída para engenharias e ciências avançadas, ou seja, uma infra-estrutura de computação que deveria prover acesso a recursos heterogêneos para aplicações com alta demanda computacional.

Outro conceito de grades é apresentado por Krauter et al. em [Krauter et al., 2002]. Segundo os autores, uma grade é um sistema computacional de rede que pode escalar para ambientes do tamanho da Internet, tendo máquinas distribuídas através de múltiplas organizações e domínios administrativos. Nesse contexto, um sistema computacional distribuído pode ser visto como um computador virtual, formado por um conjunto de máquinas heterogêneas que concordam em compartilhar seus recursos locais com os outros.

Devido à abrangência do termo, existem várias definições para uma grade computacional. No entanto, de forma geral, percebe-se que uma grade se baseia no compartilhamento de recursos computacionais heterogêneos, que estão geograficamente dispersos. Segundo Baker [Baker, 2000], existem três aspectos principais que caracterizam a computação em grade:

- *heterogeneidade*: uma grade normalmente envolve o compartilhamento de recursos heterogêneos, que podem estar dispersos por diversas plataformas e arquiteturas computacionais;
- *escalabilidade*: uma grade pode crescer de poucos para milhões de recursos compartilhados sem comprometer o bom funcionamento do ambiente;
- *dinamicidade* ou *adaptabilidade*: é preciso considerar que a probabilidade de que recursos venham a falhar é naturalmente alta; assim, os escalonadores de recursos e aplicações devem ser projetados para adaptarem o seu comportamento dinamicamente, a fim de extrair o máximo de desempenho dos recursos e serviços disponíveis.

Para que seja possível a criação de um ambiente de grade é preciso fazer uso de um dos diversos sistemas gerenciadores de recursos existentes. Esses sistemas são os responsáveis por realizarem o escalonamento e o gerenciamento dos recursos disponíveis no ambiente e das aplicações a serem computadas. Atualmente, existem várias alternativas que se destacam, entre elas cita-se o Globus [Globus, 2005], o Condor [Condor, 2005] e o OurGrid [OurGrid, 2005].

2.2 Sistemas para computação em grade

Devido às motivações promissoras apresentadas pela computação em grade, diversos sistemas gerenciadores de recursos (*Resource Management Systems* ou RMSs) têm sido desenvolvidos. Entre esses sistemas, alguns são provenientes de estudos acadêmicos (por exemplo: Globus [Globus, 2005], Condor [Condor, 2005], OurGrid [OurGrid, 2005] e Legion [Legion, 2005]) e outros, decorrentes de esforços comerciais (ex: Entropia [Entropia, 2005] e distributed.net [Distributed.net, 2005]). O projeto Globus (Sub-seção 2.2.1) é a solução que mais se destaca por buscar a padronização de serviços a serem disponibilizados em uma grade. Outras alternativas como Condor (Sub-seção 2.2.2) e OurGrid (Sub-seção 2.2.3) também apresentam características bastante interessantes. Essas três tecnologias serão apresentadas nas sub-seções a seguir.

2.2.1 Globus

O Globus [Globus, 2005] consiste em um *toolkit* de software, desenvolvido pela Globus Alliance, que pode ser usado para formar ambientes de grade e programar aplicações para serem executadas nos mesmos. Esse *toolkit* é composto por um conjunto de serviços que

implementam diferentes funcionalidades, tais como segurança, alocação de recursos, gerenciamento de dados e comunicação. Uma das características apresentadas pelo Globus é a alternativa de que se utilize esses serviços de forma independente ou em conjunto. De acordo com Cirne [Cirne, 2003a], essa possibilidade de uso parcial do Globus é um aspecto importante para a sua aceitação, pois possibilita começar utilizando funcionalidades mais básicas e, aos poucos, incorporar outras. Entre alguns dos serviços oferecidos estão:

- *Globus Resource Allocation Manager* (GRAM): fornece mecanismos para submissão, monitoramento e controle das tarefas que são executadas na grade;
- *Grid Security Infrastructure* (GSI): infra-estrutura através da qual são oferecidos serviços de segurança, tais como autenticação, controle de acesso e confidencialidade;
- *Monitoring and Discovery Service* (MDS): fornece informações sobre os recursos que compõem a grade, como disponibilidade e carga de CPU;
- *GridFTP*: protocolo para transferência de dados que estende o tradicional FTP; novas funcionalidades foram adicionadas, como transferência em paralelo (várias conexões TCP entre origem e destino) e transferência *striped* (diversas conexões TCP entre várias origens e um destino, ou vice-versa).

Outro aspecto relevante reside na quarta versão do *toolkit* (GT4), que se baseia em duas padronizações: OGSA [Talia, 2002; Foster et al., 2002] e WSRF [Foster et al., 2005]. O *Open Grid Services Architecture* (OGSA), desenvolvido pelo *Global Grid Forum* (GGF) [GGF, 2005], busca definir uma arquitetura comum, padrão e aberta para aplicações de grades. O objetivo do OGSA é padronizar praticamente todos os serviços existentes em uma aplicação de grade (por exemplo, serviços de segurança e serviços de descoberta), especificando uma série de interfaces para esses serviços [Sotomaior, 2005].

O OGSA adotou a tecnologia de serviços *web* [Ferris, 2003; Chung, 2003] para realizar as comunicações que ocorrem no ambiente de grade. Contudo, como essa tecnologia não oferecia uma forma padrão de utilizar serviços mantendo o estado da comunicação (*stateful*[3]), o OGSA adotou, também, o *Web Services Resource Framework* (WSRF). O WSRF, desenvolvido pelo OASIS [OASIS, 2005], especifica como é possível implementar serviços *web* com estado, além de determinar outras funcionalidades que os tornam mais adequados para trabalhar com aplicações de grade, padronizando, por exemplo, um meio de reportar falhas quando ocorre algo de errado durante a invocação de um serviço.

---

[3] Embora se saiba que serviços *web* podem ser implementados tanto com estado quanto sem (*stateless*), eles são comumente *stateless* e não havia um meio padrão de torná-los *stateful* [Sotomaior, 2005].

Estando baseado em padronizações como OGSA e WSRF, o projeto Globus espera alcançar uma maior interoperabilidade no uso de serviços em geral para grades computacionais, o que pode fazer com que esse tipo de computação evolua significativamente. A Figura 2.1 ilustra as relações entre GT4, OGSA, WSRF e serviços *web*. O GT4, além de uma implementação completa do WSRF (1), implementa também diversos serviços para grades (2). A maioria desses serviços é compatível com os requisitos do OGSA (3) e, também, são implementados de acordo com o WSRF (4). Enquanto o OGSA necessita de serviços *web* com estado (5), o WSRF é o responsável por especificá-los (6), os quais são uma extensão dos serviços *web* tradicionais (7).

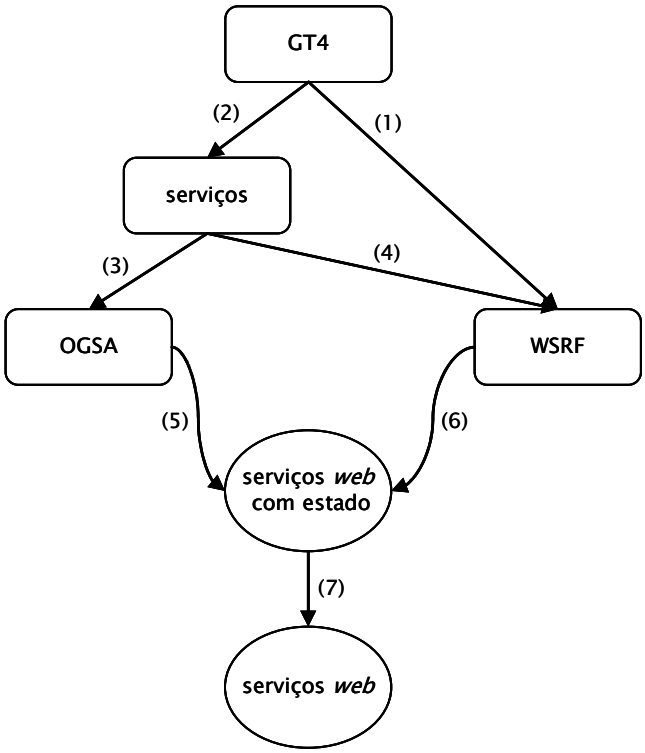
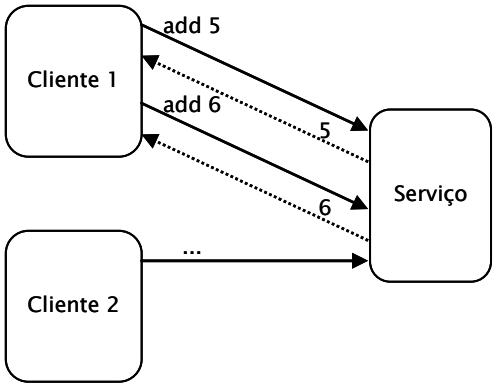


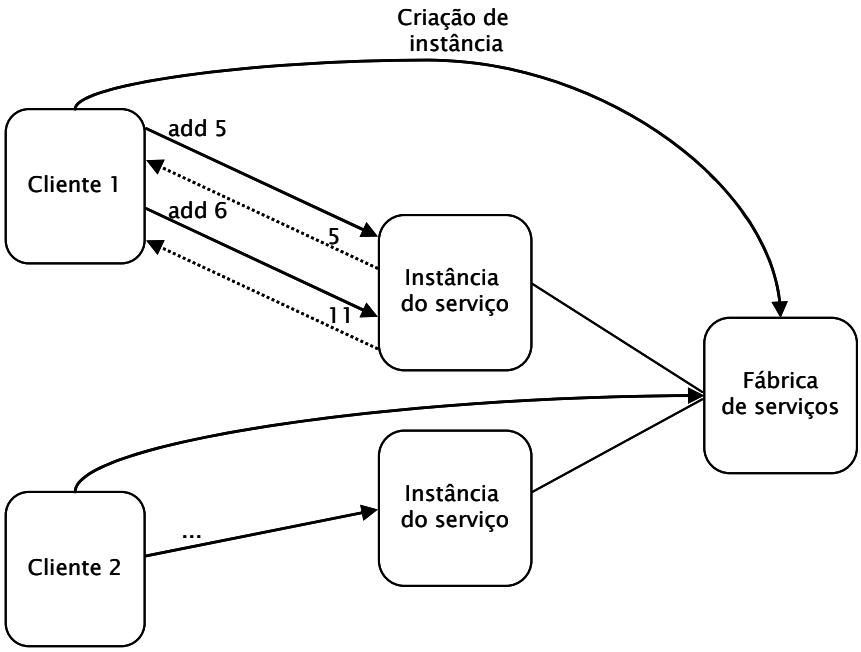
Figura 2.1. Relação entre GT4, OGSA, WSRF e serviços *web*.
Adaptada de [Sotomaior, 2005].

A Figura 2.2 ilustra um exemplo onde a diferença entre um serviço *web* tradicional pode ser percebida em comparação a uma aplicação Globus. Imagina-se, neste caso, um serviço que atua como um acumulador de números inteiros. Esse acumulador é iniciado com o valor zero e, deseja-se, então, adicionar valores ao mesmo através de uma operação chamada *add*. Na Figura 2.2 (a), é possível observar o comportamento de um serviço *web* que não mantém o estado da comunicação entre os clientes e o serviço. Na primeira invocação (*add* 5) do cliente 1, o resultado recebido é 5. A partir da segunda requisição, essas não terão conhecimento do que havia sido computado nas operações anteriores. Assim, na segunda

invocação (*add* 6) o resultado recebido é 6. Já no caso de uma aplicação Globus, como apresentado na Figura 2.2 (b), tem-se um meio padrão de implementar serviços *stateful*, sendo possível a criação de instâncias do serviço (a partir de uma fábrica de instâncias), as quais irão armazenar as informações de estado. Quando a operação *add* 5 é invocada, o resultado recebido é 5. A diferença em relação aos serviços *web* tradicionais, neste caso, ocorre a partir da segunda invocação. Percebe-se, na operação *add* 6, que o resultado recebido é 11 ao invés de 6.



(a) Serviço *web* tradicional (sem estado)



(b) Aplicação Globus (com estado)

Figura 2.2. Comparação entre um serviço *web* tradicional e uma aplicação Globus.

2.2.2 Condor

O sistema Condor [Condor, 2005], ou *Condor High Throughput Computing System*, é um gerenciador de recursos especializado para aplicações que exigem computação intensa. Produzido pelo *Condor Research Project* da Universidade de Wisconsin, seu objetivo é oferecer uma grande quantidade de poder computacional a médio e longo prazo (dias a semanas), utilizando recursos ociosos do ambiente. Os autores do sistema enfatizam que Condor objetiva alta vazão (*high throughput*) e não alto desempenho (*high performance*). Segundo Cirne [Cirne, 2003b], Condor visa fornecer desempenho sustentável a médio e longo prazo, mesmo que o desempenho instantâneo do sistema possa variar consideravelmente.

Condor apresenta algumas características relevantes. Entre elas, cita-se a habilidade de realizar a *migração de processos*. Essa migração, proporcionada por um mecanismo de *checkpointing*, consiste em salvar o estado corrente de uma tarefa em execução, permitindo que ela seja reiniciada em uma outra estação da grade do ponto em que havia parado. O uso deste mecanismo é extremamente útil, por exemplo, no caso em que uma estação venha a falhar ou tenha que ser reiniciada.

Um outro aspecto interessante apresentado pelo Condor é a capacidade de realizar o casamento (*match*) entre as requisições e as ofertas de recursos. Quando um usuário disponibiliza sua estação para fazer parte de uma grade, ele especifica, através de um arquivo de configuração, as preferências e os requisitos para que uma tarefa seja executada em sua máquina. Por exemplo, o usuário pode permitir que sua estação seja usada apenas quando estiver ociosa ou em finais de semanas. Antes de uma tarefa ser executada, é necessário que sejam encontradas estações que atendam às necessidades dessa tarefa (por exemplo, quantidade de memória disponível). Condor atua, então, com seu mecanismo chamado *matchmaker*, fazendo a negociação entre as tarefas e as estações disponíveis na grade.

Ao conjunto de estações que disponibilizam seus recursos em um ambiente Condor é dado o nome de Condor *pool*. Essas estações podem exercer diferentes funções, como apresentado a seguir:

- *central manager*: gerenciador central responsável por coletar informações da grade e fazer a negociação (*match*) entre as aplicações e os recursos; apenas uma estação do *pool* pode ser configurada como *central manager*;
- *execute machine*: todas as estações do *pool* que compartilham seus recursos para computarem aplicações;

- *submit machine*: qualquer estação do *pool* configurada com a capacidade de submeter aplicações para serem executadas; todos os arquivos de *checkpoint* são armazenados na estação a partir da qual a aplicação foi submetida;
- *checkpoint server*: função opcional que pode ser exercida por qualquer estação do *pool*; atua como um servidor centralizado para armazenar arquivos de *checkpoint* de todas as aplicações executadas na grade.

O sistema Condor é formado por um conjunto de diferentes *daemons*, que são os responsáveis por implementarem as diferentes funções que uma estação pode exercer em um Condor *pool*. Os principais deles são descritos abaixo:

- *collector*: repositório central, responsável por coletar informações sobre o estado de um Condor *pool*; atua na estação *central manager*, recebendo atualizações dos demais *daemons* (exceto do *negotiator*); essas atualizações ocorrem na forma de *ClassAd*, uma estrutura de dados baseada em um conjunto de atributos que descrevem o estado de uma entidade específica do sistema (*daemons*, recursos, aplicações);
- *negotiator*: responsável por realizar o *matchmaking* no sistema. Para isso, pergunta freqüentemente ao *collector* pelo estado corrente de todos os recursos disponíveis no *pool*;
- *startd*: executa em todas as estações que computam tarefas (*execute machines*); sua função é monitorar as condições dos recursos onde está executando e publicar *ClassAds* ofertando esses recursos; é responsável, também, por garantir que as políticas de uso, estabelecidas pelo dono dos recursos, sejam aplicadas (ex: especificar quais usuários têm preferência sobre seus recursos);
- *schedd*: executa em todas as estações a partir das quais aplicações são submetidas (*submit machines*), sendo responsável pela requisição por recursos através de *ClassAds*; quando um usuário submete uma aplicação, essa vai para uma fila gerenciada pelo *schedd*;
- *shadow* e *starter*: quando um *schedd* dispara a execução de uma aplicação, um processo *shadow* é iniciado na estação de submissão (*submit machine*), e os *daemons startd* disparam um processo *starter* nas estações de execução correspondentes (*execute machines*); o *shadow* é responsável por realizar requisições para transferência de arquivos, fazer *log* sobre a execução das tarefas e reportar estatísticas; já o *starter* é responsável por monitorar a execução da tarefa, reportando à *submit machine* quando a tarefa for concluída.

A Figura 2.3 apresenta uma visão da arquitetura de um Condor *pool*. Nela é possível perceber a presença de um *central manager* com os *daemons collector* e *negotiator*; a presença de uma *submit machine* (estação 1) com os *daemons schedd* e *shadow*; e também a presença de *execute machines* (estação 2, ..., estação n), com os *daemons startd* e *starter*. Quando uma aplicação é submetida, o *negotiator* é responsável por consultar o *collector* para fazer o *match*. O *match* é realizado entre as ofertas de recursos disponibilizadas pelos *daemons startd* e a demanda por recursos solicitada pelo *daemon schedd*. Após ter recebido a relação de recursos disponíveis, o *schedd* escalona a aplicação disparando o *shadow*. Os *daemons startd* disparam então os *daemons starter*, que irão reportar os resultados à *submit machine* quando a aplicação for concluída.

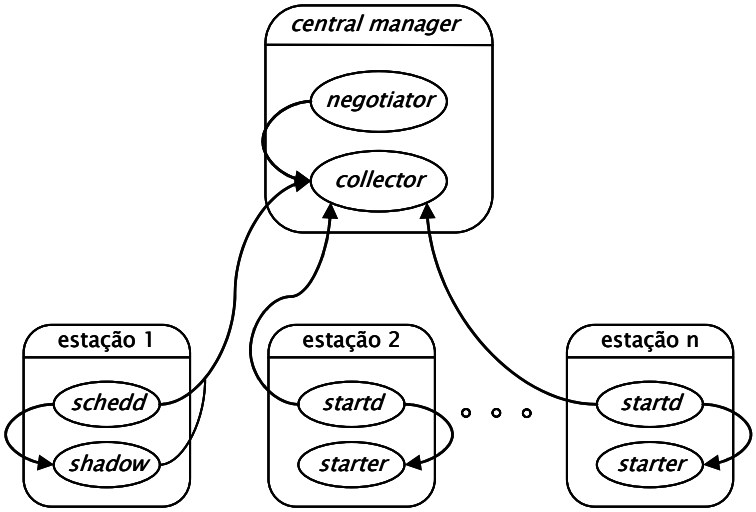


Figura 2.3. Arquitetura de um Condor *pool* com seus *daemons*.

2.2.3 OurGrid

O OurGrid [OurGrid, 2005] é uma solução de grade computacional que tem como principal objetivo a criação de um ambiente de execução para aplicações do tipo *Bag-of-Tasks* (BoT). Aplicações do tipo BoT são aplicações paralelas cujas tarefas são independentes umas das outras, ou seja, não precisam de comunicação entre si para realizar suas funções. Embora simples, são usadas em uma variedade de cenários, como *massive searches*, *data mining* e simulações Monte Carlo [Cirne et al., 2004]. O OurGrid é desenvolvido por um esforço conjunto entre a Universidade Federal de Campina Grande (UFCG) e a HP Brasil.

Uma das características principais do sistema OurGrid está relacionada ao escalonamento de tarefas. Na medida em que houver recursos disponíveis na grade, réplicas

das tarefas que estão em execução podem ser criadas com o objetivo de tentar diminuir o tempo de computação da aplicação. Assim, réplicas de uma mesma tarefa são executadas simultaneamente, sendo que a primeira a ser concluída é computada e as demais abortadas. De acordo com Cirne [Cirne et al., 2003b], apesar de o algoritmo de escalonamento não fazer uso de informações sobre as tarefas (ex: tempo estimado de computação), e tampouco dos recursos disponíveis no ambiente (ex: carga de CPU e quantidade de memória disponíveis), ele se mostra equivalente em termos de desempenho quando comparado com soluções que apresentam esse conhecimento de informações sobre o ambiente.

O sistema OurGrid pode ser descrito pelas duas camadas de software que o compõem: o MyGrid e o OurGrid. Enquanto o MyGrid se identifica por ser a grade de um único usuário, o OurGrid se caracteriza por possibilitar a troca de recursos entre diferentes domínios administrativos. Esses domínios formam uma comunidade virtual, gerando, assim, uma rede *peer-to-peer* para o compartilhamento de recursos. A troca desses recursos se dá através de uma rede de favores descentralizada [Andrade et al., 2003]: uma organização provê recursos para outra com a expectativa de ser recompensada no futuro com a execução de suas próprias tarefas remotamente. O MyGrid pode executar de forma independente de uma solução OurGrid, ou ainda, ser acoplado em um ponto OurGrid, tornando-se o *front-end* do usuário com uma comunidade.

Existem três componentes básicos que formam a infra-estrutura de grades OurGrid e que precisam ser analisados. As GuMs (*Grid Machines*) são as estações que compõem a grade, sendo as responsáveis por compartilharem seus recursos para a computação de tarefas durante a execução de uma aplicação. Para que alguma estação possa fazer parte de uma grade OurGrid, basta que um usuário ou a instituição responsável por essa grade tenha acesso à estação (transferência de arquivos e execução remota), por exemplo através de uma conexão *ssh* ou *ftp*.

Já os GuMPs (*Grid Machine Providers*) são os serviços nos quais é possível configurar quais máquinas podem ser usadas como GuMs. São também responsáveis por fornecerem GuMs dinamicamente quando solicitados por um MyGrid. Um GuMP pode, ainda, fazer ou não fazer parte de uma comunidade OurGrid.

O terceiro e último componente é o próprio MyGrid, que provê todo o suporte necessário para descrever e executar aplicações na grade. Durante a execução de uma aplicação, o MyGrid obtém GuMs, sob demanda, dos GuMPs aos quais estiver conectado. É responsabilidade do MyGrid escalonar a execução de aplicações e realizar a transferência e a coleta de arquivos para/das GuMs.

A Figura 2.4 ilustra os componentes citados acima e as relações entre os mesmos. Nela, é possível perceber a presença de um MyGrid no domínio administrativo representado por A. Esse MyGrid está conectado a um GuMP isolado (1) e, ainda, a um GuMP que faz parte de uma comunidade OurGrid (2), atuando como o *front-end* do usuário com esse OurGrid. Na nuvem é possível visualizar uma comunidade formada por três pontos OurGrid (envolvendo os domínios A, B e C). Quando o usuário dispara a execução de uma aplicação na grade, o MyGrid faz a solicitação de recursos aos GuMPs em que está conectado. Após receber uma relação de recursos disponíveis, o próprio MyGrid fica responsável por escalonar as tarefas, dando preferência pela utilização dos recursos locais.

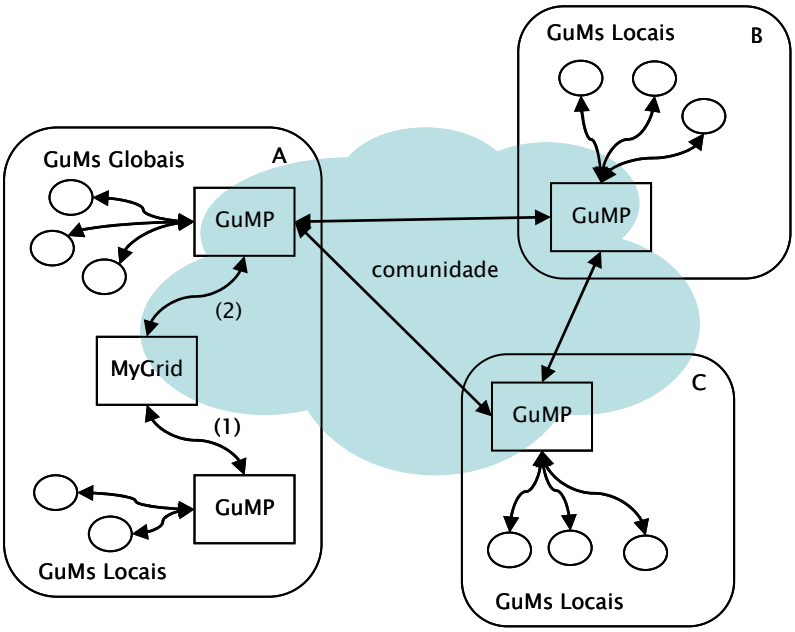


Figura 2.4. Componentes da infra-estrutura OurGrid.

2.3 Benefícios e requisitos de gerenciamento em ambientes de grade

Após ter mencionado alguns dos principais sistemas que dão suporte à computação em grade, esta seção descreve os benefícios que podem ser obtidos com o emprego de mecanismos de gerenciamento em infra-estruturas de grade, bem como os requisitos que esses mecanismos devem atender.

Atualmente, o uso de grades computacionais encontra-se em franca expansão no meio acadêmico. No entanto, em ambientes corporativos o seu emprego é mais recente, estando ainda nos primeiros estágios de exploração. Como mencionado no Capítulo 1, para que essas

grades possam ser usadas em produção, torna-se necessário dispor de mecanismos automatizados que permitam gerenciá-las eficientemente.

A incorporação de tais mecanismos pode ser extremamente útil, auxiliando, por exemplo: (a) na descoberta, manutenção e monitoração de mudanças na estrutura física e lógica da grade; (b) na detecção, isolamento e recuperação de falhas que possam vir a ocorrer; (c) no monitoramento da utilização dos recursos que são compartilhados, oferecendo medidas para a análise de desempenho e planejamento de capacidade; (d) na detecção de violações dos mecanismos de segurança que são utilizados para realizar as comunicações no ambiente; (e) na contabilização do uso da grade, proporcionando a obtenção de dados estatísticos que caracterizam a utilização da infra-estrutura, reconhecendo padrões e determinando tendências.

Ao mesmo tempo em que se deseja gerenciar ambientes de grade, a implementação desse tipo de alternativas pode trazer alguns desafios de projeto. De acordo com Massie [Massie, 2004] esses desafios estão relacionados com a construção de sistemas:

- escaláveis, ao ponto de se adaptarem ao número de nodos presentes;
- robustos, sendo capazes de sobreviver a falhas no sistema gerenciado;
- extensíveis, possibilitando que novos tipos de recursos possam ser gerenciados;
- portáveis, atuando em diferentes arquiteturas e sistemas operacionais;
- que apresentem um baixo custo de gerenciamento, evitando configurações manuais;
- que ofereçam um baixo custo adicional de consumo de recursos (como CPU, memória, E/S e banda de rede).

Percebe-se, assim, que a gerência dos recursos que compõem o ambiente bem como das aplicações que nele são executadas pode ser útil tanto do ponto de vista de usuários e administradores quanto das tecnologias de grades computacionais. Entretanto, certos cuidados devem ser levados em consideração ao se implementar tais mecanismos. Assim, espera-se que seja possível o desenvolvimento de soluções funcionais, que se adaptem às características e à dinamicidade do ambiente.

2.4 *Frameworks* de gerenciamento baseados em serviços *web*

As primeiras pesquisas na área de gerência de redes ocorreram na metade da década de 1980. Nos dias atuais, após 20 anos terem se passado, tem-se disponível um arcabouço de soluções de gerenciamento consolidado tanto no meio acadêmico quanto no meio empresarial. Mesmo assim, a busca por novos paradigmas e pelo aperfeiçoamento dessas soluções continua constantemente.

Recentemente, a tecnologia de serviços *web* [Ferris, 2003; Chung, 2003] tem evoluído como uma nova e promissora proposta para a computação distribuída. Entre suas principais características estão: (a) independência de plataformas e de linguagens de programação; (b) interoperabilidade entre sistemas; e (c) habilidade de dispor uma interface de comunicação que oculta a implementação que foi desenvolvida. De acordo com Catania [Catania et al., 2003], observou-se, a partir de então, que essa tecnologia também pode ser usada de forma eficiente na área de gerência, podendo inclusive vir a se tornar a plataforma padrão no futuro dessa área rumo a soluções mais abrangentes.

Devido às características relevantes apresentadas pelos serviços *web*, algumas propostas que integram essa tecnologia à área de gerência foram criadas. As sub-seções a seguir descrevem duas dessas propostas. Na Sub-seção 2.4.1 é apresentado o padrão *Web Services Distributed Management* (WSDM) [Vambenepe et al., 2005; Sedukhin et al., 2005] e na Sub-seção 2.4.2 é apresentado o *framework Web Services for Management* (WS-Management) [Geller et al., 2004]. Logo após, é realizada uma comparação entre as duas soluções (Sub-seção 2.4.3).

2.4.1 Web Services Distributed Management (WSDM)

O WSDM [Vambenepe et al., 2005; Sedukhin et al., 2005], projeto desenvolvido por empresas como HP, IBM e Dell em conjunto com o *OASIS WSDM Technical Committee*, tornou-se um padrão OASIS em março de 2005 e divide-se em duas especificações: (a) *Management Using Web Services* (MUWS) [Vambenepe et al., 2005]; e (b) *Management of Web Serviçes* (MOWS) [Sedukhin et al., 2005]. MUWS caracteriza-se por fazer uso da tecnologia de serviços *web* para realizar funções gerenciais, enquanto MOWS identifica-se por definir um modelo para gerenciar os próprios serviços *web*. No contexto desta dissertação, apenas o MUWS é de particular interesse e será descrito.

O MUWS tem como principal objetivo possibilitar um meio de gerenciamento fácil e eficiente, oferecendo um *framework* para gerenciar recursos através de interfaces disponibilizadas pelos mesmos. MUWS está baseado em um conjunto de especificações vinculadas a serviços *web*, como por exemplo, em serviços de descoberta [Clement et al., 2004], troca de mensagens [Gudgin et al., 2003], descrição [Booth et al., 2005] e notificação [Graham et al., 2005]. A Figura 2.5 ilustra o funcionamento básico do gerenciamento de recursos utilizando MUWS.

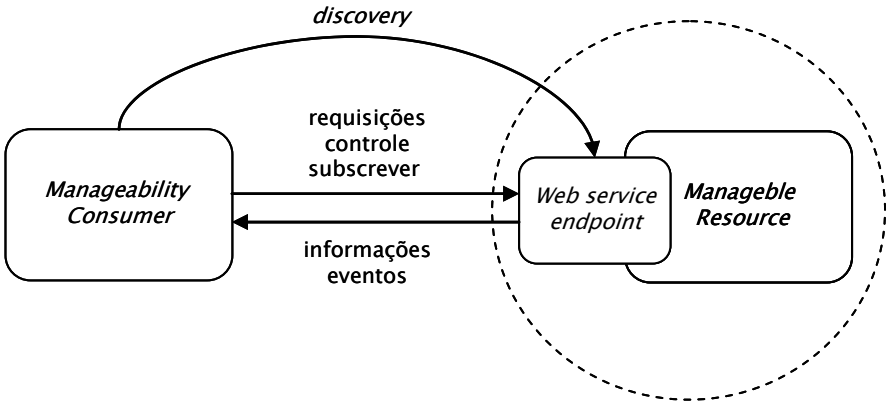


Figura 2.5. Gerenciamento de recursos através de MUWS.

O chamado *Web service endpoint* oferece acesso a um recurso gerenciável (*Manageble Resource*). Um recurso gerenciável pode ser, por exemplo, uma impressora que indica quando o nível de *toner* está baixo, ou até mesmo um disco magnético de armazenamento, que indica sua temperatura interna. Já o *Manageability Consumer* trata-se de um sistema de gerenciamento, responsável por fazer a descoberta (*discovery*) do *Web service endpoint* e trocar mensagens com o mesmo, tendo o objetivo de obter informações, subscrever por eventos, ou controlar o recurso associado.

Para que seja possível descobrir e se comunicar com um *Web service endpoint* que oferece acesso a um recurso em particular, um *Manageability Consumer* primeiramente obtém uma referência ao *endpoint*, como definido pela especificação WS-Adressing [Vinoski, 2005]. Após, o *Consumer* obtém do próprio *endpoint* uma descrição sobre o recurso gerenciável, que irá indicar como se deve proceder com a comunicação (um documento WSDL [Cerami, 2002]).

Caso um recurso gerenciável possua a capacidade de detectar a ocorrência de eventos, um modelo de notificações pode ser empregado para publicar essa ocorrência a um consumidor. Um modelo de notificação trata-se de uma forma de comunicação na qual fornecedores de serviços (*NotificationProducers*) enviam mensagens a consumidores (*NotificationConsumers*). MUWS faz uso de um mecanismo de notificação descrito pela especificação WS-Notification [Graham et al., 2005], a qual padroniza a notificação de eventos através de um modelo *publish/subscribe* baseado em tópicos (eventos). O objetivo principal do WS-Notification é padronizar os conceitos, o formato da troca de mensagens e, ainda, fornecer uma linguagem para a descrição de tópicos.

WS-Notification está sub-dividida em três especificações:

- WS-BaseNotification: define as interfaces dos serviços *web* para os *NotificationProducers* e os *NotificationConsumers* e, também, o padrão para a troca de mensagens (*NotificationMessages*);
- WS-BrokeredNotification: define a interface para o *NotificationBroker*, uma entidade opcional que quando empregada atua recebendo mensagens de publicadores de eventos (*publishers*) e é responsável por disseminá-las aos *NotificationConsumers*;
- WS-Topics: define um meio comum de organizar em categorias os itens de interesse (tópicos) para subscrição; um *subscriber*, ao fazer uma subscrição, associa um ou mais tópicos a mesma.

A Figura 2.6 exemplifica a notificação de eventos através do WS-Notification sem o uso de um *NotificationBroker*. Percebe-se a presença de um serviço *web* que desempenha o papel de um *NotificationProducer*, o qual provê suporte aos tópicos *goingOffLine* e *SystemError*. O *subscriber* (representado por A) subscreve o consumidor B pelo tópico *SystemError* e o consumidor C pelo tópico *goingOffLine* (1). Assim, quando eventos *SystemError* e *goingOffLine* forem detectados pelo *NotificationProducer*, os consumidores B e C serão notificados (2).

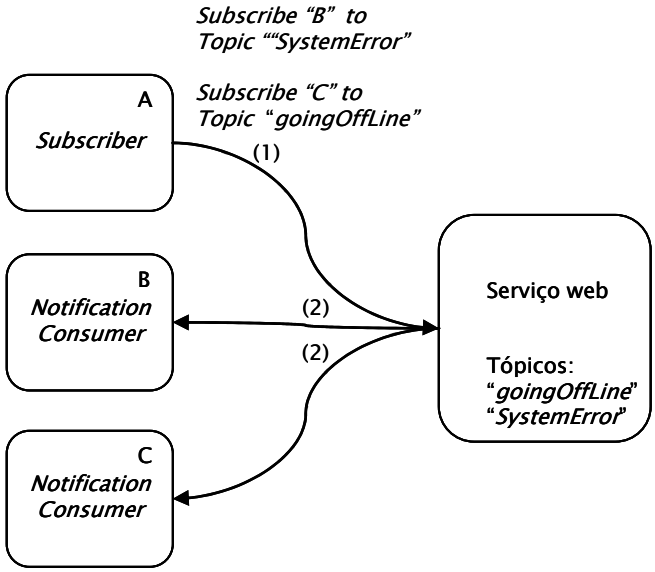


Figura 2.6. WS-Notification sem o uso de um *NotificationBroker*.

Na Figura 2.7 visualiza-se a presença de um *NotificationBroker*. Para este caso, o *subscriber* (A) subscreve o consumidor C pelo tópico *SystemError* (1). Ao receber publicações dos *publishers* relativas a esse tópico (2), o *NotificationBroker* faz a entrega das mensagens ao *NotificationConsumer* (3) subscrito.

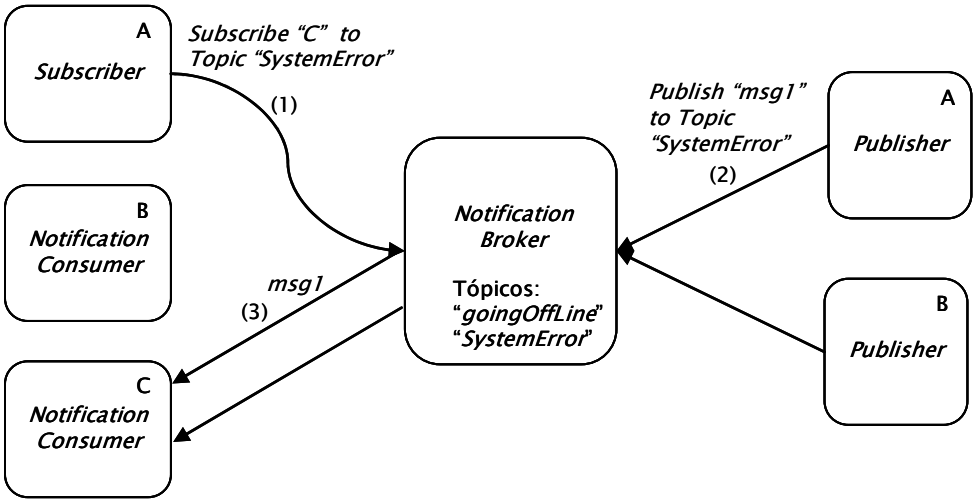


Figura 2.7. WS-Notification fazendo uso de um *NotificationBroker*.

2.4.2 Web Services for Management (WS-Management)

Outra alternativa para gerenciar recursos através de serviços *web* é o WS-Management [Geller et al., 2004]. Desenvolvido em parceria por empresas como Microsoft, Dell e Intel, o WS-Management descreve um *framework* para gerenciar sistemas como estações de trabalho, servidores, dispositivos e entidades gerenciáveis em geral. De acordo com Hamilton em [Hamilton, 2004], o WS-Management reformula o conceito de gerenciamento distribuído. Segundo o autor, um aspecto importante em sistemas distribuídos é o gerenciamento de sistemas e dispositivos. Assim, os serviços *web* oferecem uma base forte para a construção de soluções de gerenciamento robustas e interoperáveis.

A especificação WS-Management foi projetada para satisfazer alguns requisitos, entre os quais cita-se: (a) fazer uso de protocolos e especificações dos serviços *web*, para que esses serviços possam ser implementados em agentes de gerenciamento com poucos esforços tanto em hardware quanto em software; (b) assegurar agregação e composição com outras especificações de serviços *web*; e (c) minimizar a implementação de mecanismos adicionais além dos oferecidos pela arquitetura atual de serviços *web*.

O modelo apresentado pelo WS-Management possui alguns termos fundamentais. Esses são descritos a seguir:

- *System*: entidade gerenciável composta por um ou mais *Resource Instances*;
- *Resource Instance*: também chamado de *Resource* ou apenas *Instance*, trata-se de um único item gerenciável, como um *drive* de disco ou um processo em execução;

- *Resource Service*: serviço *web* que fornece acesso a uma única categoria de itens gerenciáveis (que compartilham as mesmas operações e esquema de representação), como um conjunto de *drives* de disco ou de processos em execução;
- *Agent*: aplicação que fornece serviços de gerenciamento para um *System* expondo um conjunto de *Resource Services*;
- *Manager*: serviço *web* usado para gerenciar um ou mais *Systems*.

O WS-Management identificou funções básicas de uso em sistemas de gerenciamento em geral, tendo o objetivo de fornecer em seu *framework* um conjunto de operações fundamentais de gerência. Implementações estão livres para fazer ou não uso dessas operações, o que irá depender das funcionalidades desejadas para cada entidade gerenciável. As seguintes funções são oferecidas:

- *Discover*: descobrir a presença de recursos gerenciáveis;
- *Get*, *Put*, *Create* e *Delete*: obter, postar, criar e apagar valores associados aos recursos gerenciáveis;
- *Subscribe*: subscrever para eventos emitidos por recursos gerenciáveis;

2.4.3 Considerações sobre WSDM e WS-Management

Como pôde ser observado nas duas sub-seções anteriores, tanto o WSDM quanto o WS-Management se destinam ao gerenciamento de recursos. As duas alternativas fazem uso de especificações dos serviços *web* para atingir seus objetivos.

De acordo com Kumar em [Kumar, 2004], existem três diferenças importantes entre o WSDM e o WS-Management. A primeira delas é que o WS-Management somente se preocupa com o gerenciamento de recursos utilizando serviços *web*. O WSDM, por outro lado, além de suportar o gerenciamento de recursos com serviços *web* (MUWS), também oferece uma descrição de como deve ser feito o gerenciamento dos próprios serviços *web* (MOWS).

A segunda diferença é que o WSDM adota um modelo de informações de gerenciamento com a utilização do *Web Services Resource Framework* (WSRF). O WS-Management, contudo, deixa o modelo de informações como uma escolha de critério do desenvolvedor do serviço de gerenciamento. Essa característica pode ser interessante para empresas que queiram desenvolver todo um novo sistema de gerenciamento baseado em um modelo de informações já consolidado, como o CIM (*Common Information Model*), proposto pelo DMTF (*Distributed Management Task Force*).

Por fim, a terceira diferença é que o WSDM constitui-se de um padrão aprovado pelo OASIS no ano de 2005. Já o WS-Management ainda não foi submetido para padronização. Por esse motivo o WSDM foi escolhido para ser utilizado na arquitetura de gerenciamento descrita nesta dissertação.

2.5 Sumário

Este capítulo descreveu conceitos básicos que fundamentam esta dissertação. O principal objetivo do capítulo foi apresentar um arcabouço teórico sobre o problema em questão, bem como sobre tecnologias existentes e desafios a serem superados por uma solução de contabilização e caracterização de grades computacionais.

Primeiramente, foram apresentados alguns conceitos de grades computacionais (Seção 2.1) e descritos alguns dos sistemas RMSs mais amplamente utilizados (Seção 2.2). Como se pôde perceber, o sistema Globus mostra-se bastante interessante principalmente por buscar a padronização de serviços. Já o Condor ganha destaque por seu mecanismo de *checkpointing* e por realizar um *match* entre as requisições e as ofertas de recursos. Por fim, o OurGrid apresenta um algoritmo de escalonamento baseado em replicação e está fundamentado em uma rede *peer-to-peer*, onde o compartilhamento de recursos se dá através de uma rede de favores [Andrade et al., 2003].

Logo após, comentou-se que o emprego de mecanismos de gerência é essencial para que se possa acompanhar e compreender de forma precisa o uso da grade (Seção 2.3). Foram apresentados alguns benefícios que podem ser obtidos através do uso desses mecanismos, bem como os requisitos a serem satisfeitos pelos mesmos para que sejam realmente funcionais. Por fim, dois *frameworks* que podem ser empregados para o gerenciamento de recursos utilizando a tecnologia de serviços *web* foram descritos (Seção 2.4).

3 Trabalhos relacionados

Este capítulo apresenta uma revisão bibliográfica dos principais trabalhos relacionados a esta proposta. Da Seção 3.1 à Seção 3.7 são descritas algumas iniciativas que envolvem gerenciamento de ambientes de grade, a saber: visPerf, Grid Monitoring Architecture, Grid Resource Monitoring, Resource Monitoring System, MonALISA, Ganglia e Usage Record e Resource Usage Service. Já na Seção 3.8 são apresentadas considerações gerais sobre os trabalhos que foram apresentados.

3.1 visPerf

Lee, em sua ferramenta visPerf [Lee, 2003], preocupa-se em monitorar grades fornecendo visualizações sobre o estado da utilização dos recursos. Entre suas funcionalidades cita-se a coleta de dados para medidas de desempenho (carga de CPU, uso de memória, operações de entrada e saída, etc.). Originalmente, visPerf foi projetada para a solução de grades NetSolve [NetSolve, 2005], contudo já pode ser utilizada por outras tecnologias, como Globus [Globus, 2005], Condor [Condor, 2005] e Legion [Legion, 2005].

A Figura 3.1 ilustra a arquitetura de visPerf. Nessa figura é possível observar a presença de diferentes domínios administrativos que formam uma grade (representados por A, B e C) e a interação entre os três componentes que compõem a arquitetura: *visSensors*, *visPerf* e *Monitor Directory Service* (MDS). O MDS trata-se de um serviço de diretório usado para o anúncio e a descoberta dos *VisPerfs* (1). O *visPerf* é um controlador central (único por domínio administrativo) responsável por receber as informações de monitoração provindas dos *visSensors* (2), analisá-las e apresentá-las aos usuários. Os *visSensors* são sensores que executam nas máquinas que formam a grade coletando informações das mesmas. Lee faz uso de uma topologia *peer-to-peer* para formar o sistema de monitoração e permitir a troca de informações entre diferentes domínios administrativos (3), tornando possível gerenciar os domínios locais de forma centralizada e a grade inter-institucional de forma descentralizada.

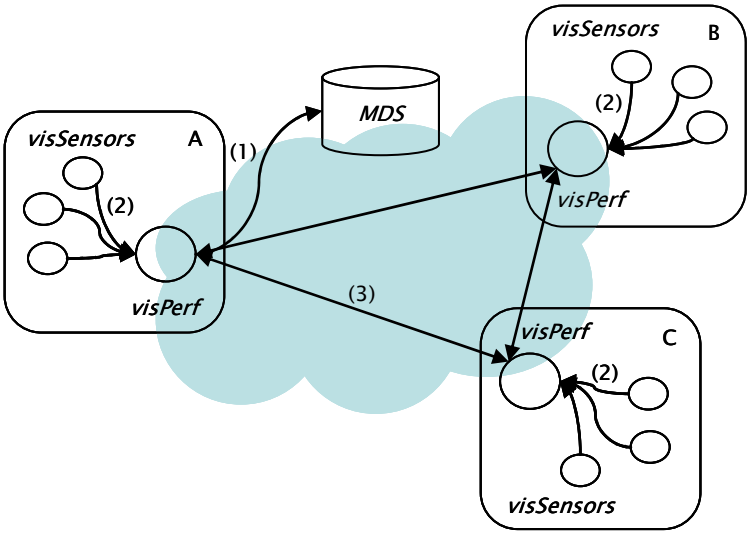


Figura 3.1. Arquitetura de VisPerf e a interação entre seus componentes.

A transferência de dados ocorre na forma de um modelo *publish/subscribe*, onde o *subscriber* é o *visPerf*, e os *publishers* são os *visSensors*. Para a entrega dos dados monitorados dois métodos podem ser utilizados: *push* e *pull*. Se um *visSensor* estiver configurado para transferir dados para um *visPerf*, o método *push* é utilizado. Caso um *visPerf* deseje obter os dados monitorados sob demanda, é enviada uma requisição aos *visSensors* usando o método *pull*.

3.2 Grid Monitoring Architecture (GMA)

Tierney define em [Tierney, 2002] uma arquitetura para o monitoramento de recursos em ambientes de grade, chamada *Grid Monitoring Architecture* (GMA). O autor buscou apresentar em seu trabalho o mínimo de especificações necessárias para dar suporte às funcionalidades de monitoração e, ainda, permitir a interoperabilidade entre ferramentas de gerência.

Como pode ser visto na Figura 3.2, GMA está baseada em um modelo produtor/consumidor, sendo formada por três componentes: o produtor (*producer*), o consumidor (*consumer*) e o serviço de diretório (*directory service*). O serviço de diretório atua como um repositório central, capaz de suportar o anúncio e a descoberta de produtores e consumidores (1). Contudo, ele não é responsável pelo armazenamento dos dados de monitoração, guardando apenas informações sobre com quais tipos de eventos um produtor ou consumidor está apto a trabalhar. Tanto consumidores quanto produtores podem fazer uso do serviço de diretório para descobrirem uns aos outros.

A transferência dos dados monitorados pode ocorrer por meio de três modelos de interação: *publish/subscribe*, *query/response* ou *notification*. Independente de quem comece a comunicação, a transferência ocorre diretamente entre o par produtor/consumidor (2), sem envolvimento adicional do serviço de diretório.

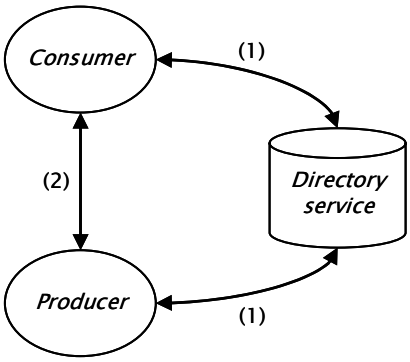


Figura 3.2. Componenentes da arquitetura GMA e suas interações.

3.3 Grid Resource Monitoring (GridRM)

A proposta de Baker [Baker, 2003] é um sistema extensível e genérico, chamado GridRM, que também está voltado à monitoração da utilização de recursos em grades. Entretanto, o principal objetivo desse sistema é coletar informações sobre recursos em diversos ambientes. Para isso, são utilizados agentes que atuem de acordo com as necessidades de cada um desses ambientes como, por exemplo, SNMP (v1, v2, v3) e Network Weather Service (NWS) [NWS, 2005].

Como ilustra a Figura 3.3, a arquitetura GridRM é composta por *gateways* (*GridRM Gateways*), agentes (*agents*) e um serviço de diretório (*directory service*). Os agentes são responsáveis por monitorar os recursos e repassar as informações obtidas ao *gateway* (1). O *GridRM Gateway* é usado para coordenar os agentes coletores em cada domínio (representados por A, B e C) e, ainda, fazer o controle de acesso aos dados. Aplicativos cliente podem se conectar a qualquer *Gateway* a que tenham autorização para obter as informações de interesse. As requisições para dados remotos são desviadas primeiramente para o serviço de diretório que, além de ser responsável pelo anúncio e descoberta dos *GridRM Gateways* que fazem parte da grade (2), endereçará o pedido ao *Gateway* que possui as informações desejadas. O *directory service* usado por GridRM é o mesmo proposto pela arquitetura GMA (Seção 3.2).

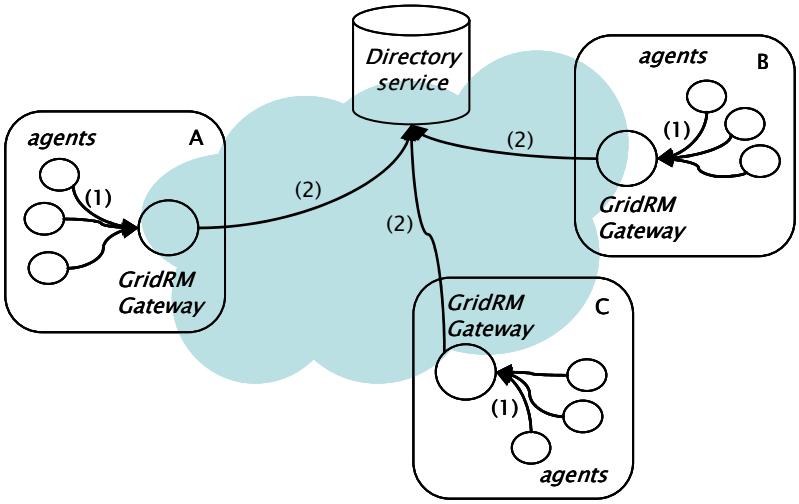


Figura 3.3. Arquitetura GridRM e a interação entre seus componentes.

Baker faz uso de um modelo *publish/subscribe*, no qual as informações coletadas pelos vários tipos de agentes são armazenadas de acordo com um modelo de dados uniforme. A intenção do autor é oferecer uma visão homogênea dessas informações.

3.4 Resource Monitoring System (Remos)

Outro trabalho relacionado é o sistema Remos, proposto por Dinda [Dinda, 2001], que se destina a monitorar recursos em sistemas distribuídos em geral. Remos coleta informações em diferentes tipos de redes e arquiteturas, utilizando diversos tipos de coletores. Para tal, esses coletores fazem uso de diferentes tecnologias, assim como SNMP e *benchmarking*.

Como é possível observar na Figura 3.4, a arquitetura Remos é composta por três componentes: *modelers*, *collectors* e *predictors*. Os *modelers* são os responsáveis por solicitar as informações de monitoração ao *Master Collector* (1) e por fazerem o processamento necessário para uniformizar essas informações. Os *collectors* são os responsáveis por obter os dados dos recursos monitorados e repassá-los ao *Master Collector* quando requisitados (2). O *Master Collector* possui uma base de dados sobre a localização de todos os outros coletores dispersos pela rede, ocultando ao *modeler* que a pesquisa foi realizada por vários coletores. Já os *predictors* são os responsáveis por consultar os *modelers* para transformar um histórico de medidas em uma predição de comportamento futuro (3). As interações entre os componentes são realizadas através do modelo *query/response*.

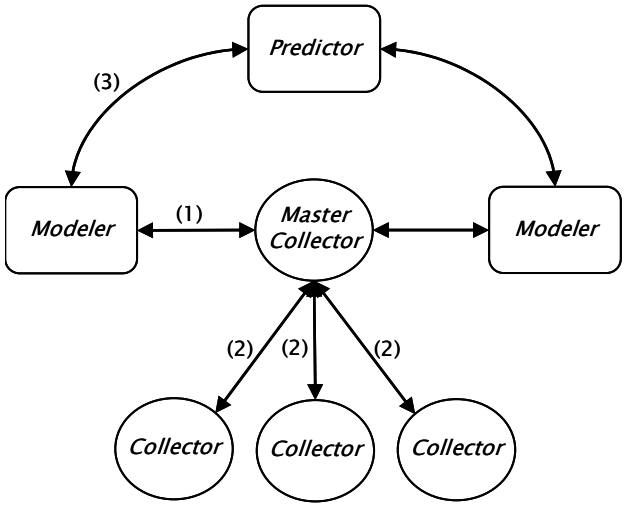


Figura 3.4. Arquitetura Remos e a interação entre seus componentes.

3.5 MonALISA

MonALISA (*Monitoring Agents in a Large Integrated Services Architecture*) [Newman, 2003] constitui-se de um *framework* para a monitoração de recursos em ambientes distribuídos. Esse *framework* foi desenvolvido sobre a *Dynamic Distributed Services Architecture* (DDSA), uma arquitetura projetada para satisfazer as necessidades de pesquisadores da área de física no monitoramento de grades computacionais.

A arquitetura DDSA, como ilustra a Figura 3.5, disponibiliza um conjunto de serviços chamados *MonALISA Services*. Esses serviços são posicionados de forma estratégica (um por instituição participante da grade, representadas por A e B) e têm a função de receber as informações coletadas por uma variedade de agentes (*agents*) (1). Os dados obtidos são armazenados localmente em uma base de dados relacional, usando um modelo normalizado de informações (2). Os *MonALISA Services*, implementados na forma de serviços *web*, são registrados em um serviço de *lookup* (*lookup service*) (3) para poderem ser descobertos por aplicativos clientes que necessitem das informações que são oferecidas (4). Após a fase de descoberta, os clientes estão aptos a consultar os *MonALISA Services* de interesse (5). A DDSA também suporta um modelo de subscrição para um conjunto de eventos, para que, assim, clientes possam ser automaticamente notificados da ocorrência dos mesmos.

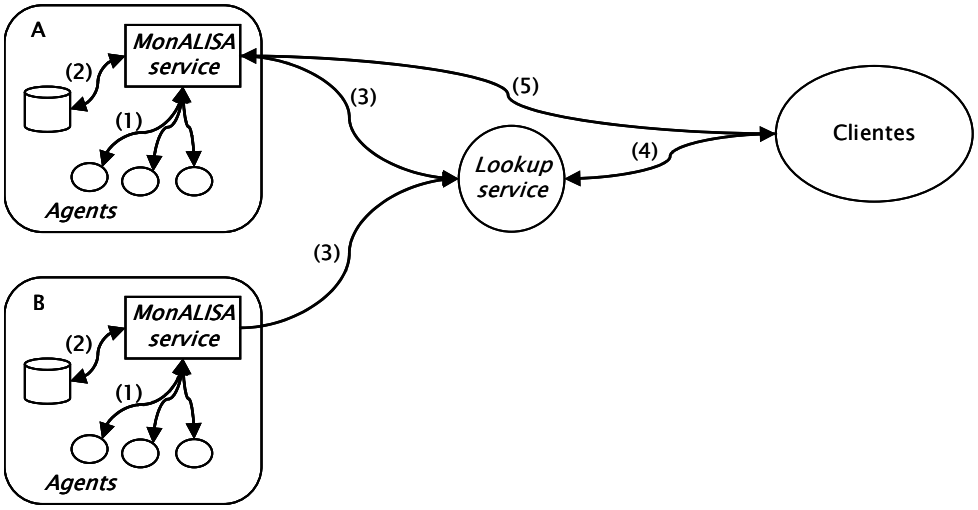


Figura 3.5. Interação entre componentes da arquitetura de MonLISA.

Os *agents* monitoram estações, roteadores e *switches* utilizando agentes SNMP. Além disso, MonALISA possibilita a integração de algumas soluções de monitoração para coletar parâmetros que descrevem o estado dos recursos, como MRTG [MRTG, 2005] e Ganglia [Massie, 2004]. Uma interce gráfica também é oferecida. Essa interface é usada para fazer a comunicação com os serviços dos quais o usuário deseja obter informações e, ainda, para gerar gráficos dos resultados obtidos. Informações em tempo real e históricas podem ser visualizadas.

3.6 Ganglia

Massie apresenta em [Massie, 2004] a solução Ganglia, um sistema distribuído destinado a monitorar sistemas de computação que visam alto desempenho, como *clusters* e grades computacionais. Dois diferentes grupos de métricas podem ser monitoradas: (a) as pré-definidas, formadas por métricas coletadas pelo próprio sistema, como porcentagem de uso de CPU e de memória; e (b) as definidas pelos usuários, que são informações arbitrárias coletadas por programas externos e incorporadas à Ganglia através de uma interface específica.

A Figura 3.6 ilustra a arquitetura de Ganglia e as relações entre seus componentes. Como pode ser visto, Ganglia é baseado em uma arquitetura hierárquica focada em federações de *clusters* (representadas por A e B), formando uma árvore de conexões *peer-to-peer*. A implementação consiste em dois *daemons*: *gmond* e *gmetad*. O *Ganglia monitoring daemon* (*gmond*) fornece dados de monitoração em um *cluster* através de um protocolo de escuta/anúncio. Sendo executado em cada nodo que se deseja monitorar, o *daemon* anuncia

mudanças significativas no estado dos recursos e responde a requisições, fornecendo os resultados em formato XML (1). Já o *Ganglia meta daemon* (*gmetad*) é responsável por receber e analisar os dados XML coletados. O *gmetad* provê ainda a integração de informações oriundas de múltiplos *clusters* através de uma árvore de conexões entre vários *daemons gmetad* (2). Para o armazenamento e a visualização das informações, Ganglia utiliza o aplicativo cliente RRDToll (*Round Robin DataBase*) [RRDToll, 2005] (3).

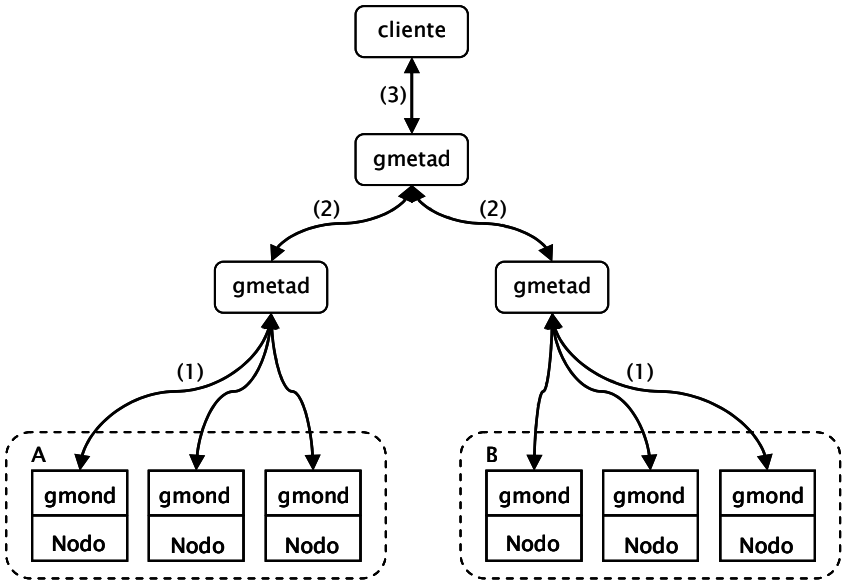


Figura 3.6. Interação entre os componentes da arquitetura de Ganglia.

3.7 Usage Record (UR) e Resource Usage Service (RUS)

Atualmente, diferentes sistemas de grades computacionais empregam formas próprias para representar indicadores sobre as aplicações executadas e os recursos consumidos, o que acaba dificultando a interoperabilidade entre instituições e ferramentas. O UR [UR, 2005], desenvolvido pelo *Usage Record Working Group* (UR-WG) do *Global Grid Forum* [GGF, 2005], surgiu da idéia de que um registro de formato uniforme deveria ser criado para armazenar essas informações. A principal motivação desse registro seria facilitar a troca de informações entre diferentes aplicações e domínios administrativos, estabelecendo um meio comum de representar os dados que estão sendo compartilhados.

A Tabela 3.1 enumera os campos presentes no UR. Como é possível observar, o modelo agrega informações relacionadas ao consumo de recursos (ex: *Processors* e *NumNodes*) e à execução das aplicações (ex: *JobName* e *Status*).

Tabela 3.1. O modelo de informações Usage Record.

Nome	Descrição
UserName	*Login* do usuário correspondente à identificação no arquivo /etc/passwd
ProjectName	Nome do projeto ou grupo
JobId	Identificação do *job*[4] submetido à fila de escalonamento
Queue	Nome da fila para a qual o *job* foi submetido
GridId	Identificador único global do usuário
FromHost	Nome da estação a partir da qual o *job* foi escalonado
ExecHost	Nome da estação que executou o *job*
StartTime	Data e hora em que o *job* começou a execução (UTC timezone)
EndTime	Data e hora em que o *job* completou a execução (UTC timezone)
Processors	Número de processadores utilizados
NumNodes	Número de nodos utilizados
CpuTime	Tempo de processamento utilizado somando todos os processos do *job*
WallTime	Tempo em que o *job* esteve no estado *running*
Memory	Quantidade máxima de memória virtual utilizada por todos os processos do *job*
Disk	Espaço de armazenamento em disco utilizado
Network	Banda de rede utilizada pelo *job*
JobName	Nome do *job* ou aplicação
Status	Estado final da execução do *job*
Charge	Valor a ser cobrado pela execução do *job*

É importante ressaltar, que o UR-WG não está interessado na forma utilizada para a coleta dos dados e tampouco em como esses dados serão utilizados. Ainda, destaca-se que são previstas extensões do modelo para atributos não identificados ou para novos atributos que possam surgir ao longo do tempo. Ademais, não se torna obrigatória a utilização de todos os campos previstos, já que uma solução de grade particular pode não fazer uso de todos esses campos.

Para agregar funcionalidades ao UR, surgiu o Resource Usage Service (RUS) [RUS, 2005], trabalho realizado pelo *Resource Usage Service Working Group* também do *Global Grid Forum* [GGF, 2005]. O RUS oferece um serviço *web*, implementado de acordo com o padrão OGSA [OGSA, 2005], para o armazenamento de registros que estão relacionados com o consumo de recursos em uma grade Globus (fazendo uso do modelo UR). Duas funções são oferecidas: (a) inserção de informações no modelo UR sobre o uso de recursos; e (b) recuperação dos dados que foram armazenados.

O objetivo do RUS é fornecer uma infraestrutura básica, que possa dar suporte à auditoria e ao monitoramento dos recursos consumidos pelos serviços executados na grade. As informações oferecidas podem ser usadas por diversas entidades, por exemplo: (a) um administrador de grade que deseja auditar a utilização dos recursos consumidos; e (b) uma

[4] A palavra *job* irá se referir, nesta dissertação, a uma unidade de execução (indivisível) manipulada pelo escalonador.

instituição que deseja cobrar pelo uso dos recursos consumidos em seu domínio administrativo.

3.8 Considerações gerais

Este capítulo apresentou um estudo que revelou a existência de importantes iniciativas relacionadas a esta dissertação. Pôde-se perceber, de maneira geral, que essas iniciativas apresentam o objetivo de monitorar a utilização de recursos em um ambiente de grade, diferenciando-se principalmente no modelo das arquiteturas apresentadas. A Tabela 3.2 oferece uma visão esquemática e comparativa desses trabalhos.

Tabela 3.2. Comparação entre os trabalhos relacionados a esta proposta.

	Escopo da monitoração	Controle seletivo de divulgação das informações	Conformidade com padrões	Modelo usado para transferência de informações
visPerf	Recursos	Não	XML	*publish/subscribe*
GMA	Recursos	Não	Não	*publish/subscribe*
GridRM	Recursos	Não	XML; SNMP	*publish/subscribe*; *query/response*; *Notification*
Remos	Recursos	Não	XML; SNMP	*query/response*
MonALISA	Recursos e aplicações	Não	XML; SNMP	*publish/subscribe*; *query/response*
Ganglia	Recursos	Não	XML	*listen/announce*; *query/response*
UR	Recursos e aplicações	Não	-	-

Como pode ser observado, os trabalhos analisados apresentam algumas limitações. A grande maioria está preocupada unicamente em monitorar a utilização dos recursos presentes na grade, desconsiderando informações estatísticas sobre as aplicações que são computadas. MonALISA e UR apresentam alguns diferenciais em relação às outras soluções ao lidarem também com dados sobre as aplicações executadas no ambiente. No entanto, MonALISA falha por não oferecer um meio através do qual seja possível vincular as informações de aplicações com as informações sobre o consumo de recursos, o que pode impedir uma

compreensão mais profunda sobre o uso da infra-estrutura. UR, por sua vez, se resume a um modelo de informação que inclui dados sobre aplicações executadas e recursos consumidos na grade. Por não constituir uma arquitetura, UR não se preocupa com a forma como esses dados devem ser coletados, processados, consolidados e apresentados ao administrador da grade.

No que se refere à divulgação seletiva de informações, alguns poucos sistemas, como visPerf [Lee 2003] e GridRM [Baker 2003], fornecem mecanismos de controle de acesso às informações coletadas. Esses mecanismos, contudo, são bastante limitados. A solução visPerf, por exemplo, faz uso de uma credencial única na grade, ou seja, todos os usuários que possuem essa credencial têm acesso a todas as informações que são geradas. Já a solução GridRM apresenta um mecanismo de controle de acesso por domínio administrativo. Quando uma instituição resolve fazer parte de uma grade que utiliza GridRM, essa passará a divulgar, aos domínios que lhe interesse, todas as informações geradas pelos seus agentes de monitoração. Em síntese, o que se observa em ambas as soluções é a pouca flexibilidade – tudo ou nada – na definição de políticas de divulgação das informações.

Em relação ao quesito de conformidade com padrões, observa-se que a maioria das soluções não adotam um padrão da área de gerência de redes, optando por alternativas proprietárias. O que mais se observa na direção de implementação de serviços interoperáveis é o uso de XML, que se resume na formatação das mensagens que são trocadas. Outras poucas soluções fazem uso de SNMP como padrão de gerenciamento, o que agrega valor no sentido de uma solução interoperável.

Por fim, quanto ao modelo utilizado para a transferência de informações, verifica-se que algumas alternativas são mais flexíveis e outras menos. GridRM, MonALISA e Ganglia, por exemplo, oferecem a opção de consulta por informações quando desejado e também de notificação à medida em que eventos ocorrem. Por outro lado, visPerf e GMA somente oferecem informações através de notificações.

4 Grid Usage Accounting and Characterization Service

A incorporação de mecanismos de gerência em ambientes de grade é de extrema relevância (Seção 2.3). Apesar de já existirem iniciativas com esse objetivo (Capítulo 3), essas apresentam algumas carências (Seção 3.8). Nesse contexto, desenvolveu-se uma arquitetura para a contabilização e a caracterização do uso de grades computacionais que busca suprir as limitações das alternativas existentes. A arquitetura proposta traz como principais contribuições:

- a contabilização e a caracterização do uso global de grades computacionais em ambientes formados por diferentes tecnologias de grade;
- o suporte para a geração de informações estatísticas sobre as aplicações executadas e sobre o consumo de recursos, visando a associação dessas informações entre si;
- um esquema seletivo para a divulgação das informações que são geradas;
- a conformidade com o padrão de gerenciamento WSDM [Vambenepe et al., 2005; Sedukhin et al., 2005].

O trabalho que aqui será apresentado é decorrente do amadurecimento de um estudo preliminar realizado durante o ano de 2004. Os resultados desse estudo podem ser vistos em [Ludwig, 2005a; Ludwig, 2005b]. O restante deste capítulo está organizado da seguinte forma: a Seção 4.1 relata os requisitos da abordagem. Na Seção 4.2 apresenta-se uma visão conceitual da arquitetura proposta, e na Seção 4.3 são descritos os componentes que compõem essa arquitetura. Finalizando, na Seção 4.4 são feitas considerações sobre o protótipo desenvolvido.

4.1 Requisitos

Em um ambiente de grade computacional, as informações de interesse podem encontrar-se espalhadas por diversas instituições e, ainda, serem geradas por diferentes sistemas gerenciadores de recursos, como Globus [Globus, 2005], Condor [Condor, 2005] e OurGrid [OurGrid, 2005]. Como cada sistema tende a empregar uma forma própria para representar indicadores sobre as aplicações executadas e os recursos consumidos, surge a

necessidade de implementação de uma arquitetura extensível e genérica, capaz de permitir a contabilização e a caracterização em ambientes formados por diferentes e diversos sistemas.

Um requisito não menos importante para uma solução de gerência de contabilização e caracterização de uso de grades computacionais é que a mesma ofereça informações estatísticas não apenas sobre o uso dos recursos que compõem a infra-estrutura, mas também sobre as aplicações que são executadas, de forma que seja possível associar a computação dessas aplicações com os recursos consumidos. Dessa maneira, torna-se possível obter uma visão mais precisa e detalhada sobre o uso da grade.

Um outro aspecto fundamental a ser levado em consideração é a questão de privacidade das informações. Em alguns casos, por exemplo, pode ocorrer que uma determinada organização não possa divulgar os dados de contabilização para certas instituições participantes da grade (por estar infringindo as políticas de segurança que regem sua instituição) ou, ainda, deseje selecionar quais dessas informações está disposta a compartilhar (por exemplo, somente divulgar dados sobre o tempo de computação de uma aplicação e não informar em quais estações essa aplicação executou). Verifica-se, assim, a necessidade de um mecanismo capaz de controlar de forma seletiva a divulgação das informações geradas.

Ressalta-se, também, que uma solução deve procurar fazer uso dos padrões de gerenciamento existentes, facilitando a interoperação entre soluções fornecidas por diferentes fabricantes. A idéia central é que a partir de plataformas de gerenciamento, tais como HP OpenView [OpenView, 2005] e IBM Tivoli [Tivoli, 2005], seja possível, por exemplo, interagir e disparar procedimentos gerenciais envolvendo a arquitetura proposta.

Por fim, deve ser dada importância à quantidade de instituições e de recursos que podem compor a infra-estrutura. Problemas de escalabilidade podem vir a ocorrer devido ao grande volume de informações que pode ser gerado pela grade. Observa-se, aqui, a necessidade de uma arquitetura descentralizada para fazer a computação dessas informações, de forma a minimizar a ocorrência de possíveis problemas.

Um cenário típico onde se faz necessário uma solução que atenda esses requisitos pode ser visualizado na Figura 4.1. Essa figura ilustra o exemplo de um ambiente de grade formado por três instituições (representadas por A, B e C), onde se percebe as relações entre elas e a diversidade de sistemas RMSs envolvidos. Na instituição A visualiza-se a presença de um OurGrid *peer* (GuMP) e um cliente Globus, além de estações aptas a fornecerem recursos para essas duas soluções. Já no domínio B, observa-se um OurGrid *peer* (GuMP) e um Condor *scheed*, com estações disponíveis para computarem tarefas provindas desses sistemas.

Nota-se, em (1), que as instituições A e B compartilham recursos através de seus OurGrid *peers*. Por fim, no domínio C, tem-se a presença de um cliente Globus e estações disponíveis para trabalhar com soluções Globus e Condor. Em (2) percebe-se o cliente Globus da instituição A computando tarefas em C. O mesmo acontece em (3), onde uma tarefa é disparada pelo Condor *schedd* presente no domínio B.

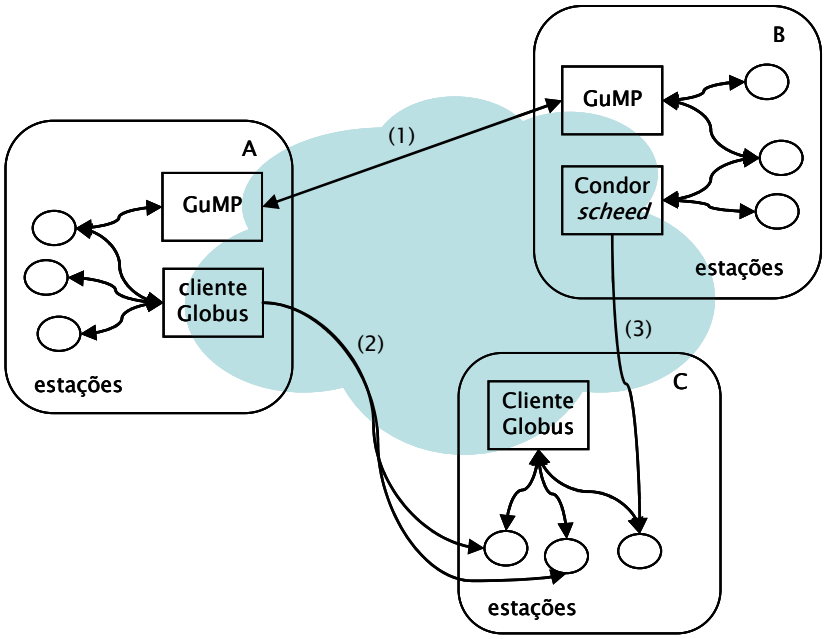


Figura 4.1. Grade formada por instituições que fazem uso de diferentes tecnologias.

4.2 Visão conceitual

Esta seção descreve a arquitetura proposta para a contabilização e a caracterização do uso de grades computacionais. A arquitetura é constituída por quatro componentes: (a) serviço de contabilização e caracterização, (b) *publishers*, (c) serviço de autorização e (d) aplicação de gerenciamento. A Figura 4.2 ilustra uma visão geral da arquitetura, instanciada em uma infra-estrutura de grade composta por dois domínios administrativos distintos: A e B. Esses domínios compartilham recursos – representados na figura por estações – empregando escalonadores de duas tecnologias diferentes de grade (E_1 e E_2).

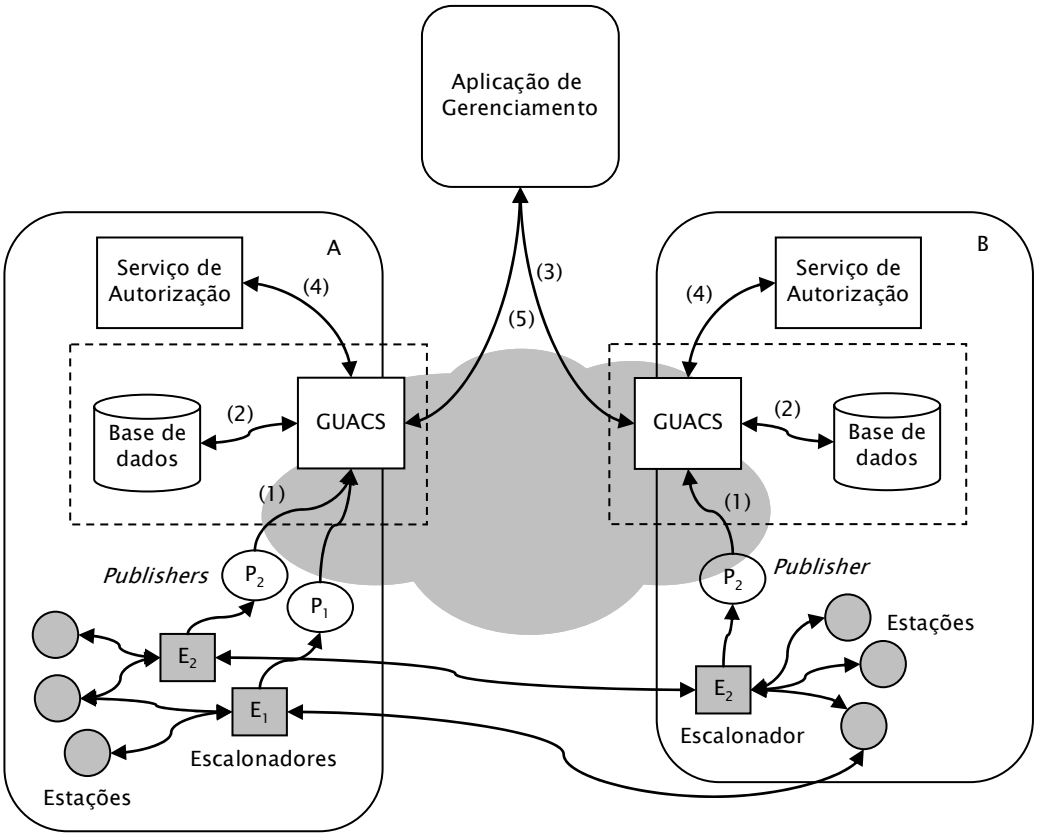


Figura 4.2. Visão geral da arquitetura e sua instanciação em ambiente de grade.

O serviço de contabilização e caracterização, denominado GUACS (*Grid Usage Accounting and Characterization Service*), atua consolidando dados gerados por *publishers* localizados na parte da infra-estrutura sob sua tutela e repassando informações sob demanda a uma ou mais aplicações de gerenciamento.

GUACS recebe dados oriundos dos *publishers* e os armazena em uma base de dados local normalizada (fluxos 1 e 2 na Figura 4.2). Por *normalizada* entenda-se que, independente do sistema de grade sendo monitorado, as informações armazenadas são as mesmas (i.e. modelo de informação é o mesmo). Para minimizar possíveis problemas com escalabilidade, o serviço pode ser instanciado de acordo com o tamanho e a complexidade da infra-estrutura (ex: por instituição ou por departamento).

Além de receber e consolidar dados oriundos dos *publishers* e armazená-los na base de dados, GUACS é responsável por servir a requisições por informações históricas e/ou em tempo "quase real" sobre os recursos consumidos e as aplicações executadas na grade. Essas requisições, originadas a partir de uma aplicação de gerenciamento (fluxo 3 na Figura 4.2), podem ser de dois tipos: *publish/subscribe* e *query/response*.

Os *publishers* são agentes de software responsáveis por monitorar a infra-estrutura de grade em tempo real aguardando pela ocorrência de eventos pré-definidos (ex: uma mensagem de *log* informando que uma aplicação foi submetida para execução). Sempre que um evento é observado, o *publisher* obtém as informações de interesse, normaliza as mesmas e as envia ao serviço GUACS.

Os *publishers* devem ser implementados e distribuídos para atuarem de acordo com as necessidades de cada tecnologia de grade, já que cada sistema possui uma forma própria para representar indicadores sobre a execução de aplicações e o consumo de recursos. Voltando ao exemplo ilustrado na Figura 4.2, suponha que o cenário é composto por Globus (representado pelo escalonador E_1) e OurGrid (escalonadores E_2). Nesse caso, *publishers* específicos para Globus e OurGrid, representados na figura por P_1 e P_2, são instanciados nos domínios, nos locais em que dados sobre recursos e aplicações são gerados.

O terceiro componente da arquitetura, que atua de forma bastante próxima ao GUACS, é o serviço de autorização. Esse serviço tem por objetivo permitir a definição e a verificação de políticas de divulgação das informações que são mantidas por GUACS. Ao receber um pedido de subscrição para um determinado evento, ou, ainda, uma consulta por dados históricos (fluxo 3 na Figura 4.2), GUACS invoca o serviço de autorização (4) que, com base em um mecanismo de controle de acesso baseado em papéis, determina se o pedido pode ser atendido ou não. Dependendo da requisição realizada e da instituição solicitante, é enviada uma resposta (*i*) reportando que o pedido foi negado ou (*ii*) com informações completas ou parciais (5).

A aplicação de gerenciamento constitui o quarto, e último, componente da arquitetura. A partir dela, o administrador pode subscrever por/ser notificado de eventos gerados pela infra-estrutura de grade. Adicionalmente, pode solicitar informações históricas de uma ou mais instâncias de GUACS e consolidá-las, visando a contabilizar e caracterizar o uso da grade. Note que, graças ao emprego de um modelo de informação normalizado, todas as informações obtidas junto às instâncias de GUACS possuem o mesmo formato, independente do sistema de grade usado em cada instituição. Portanto, é possível gerar uma visão integrada e uniforme de uso da grade. Naturalmente, essa visão pode ser comprometida se a política de distribuição de informações aplicada em cada instituição for demasiadamente restritiva.

É possível ter várias instâncias da aplicação de gerenciamento executando simultaneamente (ex: uma para cada domínio ou departamento). Cada uma delas apresentará uma visão mais ou menos detalhada da infra-estrutura de grade, dependendo das permissões que o requisitante tiver junto às instâncias de GUACS envolvidas.

4.3 Componentes

Esta seção detalha os componentes da arquitetura recém introduzida. Ressalta-se que a arquitetura é fundamentada no padrão WSDM [Vambenepe et al., 2005; Sedukhin et al. 2005]. Nesse contexto, outras especificações associadas como o WS-Notification [Graham et al., 2005] e o WSRF (*Web Service Resource Framework*) [Foster et al., 2005] também são intensivamente utilizadas.

4.3.1 Serviço de Contabilização e Caracterização

O serviço de contabilização e caracterização opera como um componente intermediário entre os *publishers* e a aplicação de gerenciamento. Dois elementos são chave nesse serviço: o modelo de informação para armazenar as informações sobre o uso da grade e a interface WSDM para comunicação com *publishers* e aplicação de gerenciamento.

Modelo de Informação

Como mencionado na Seção 4.2, adotou-se um modelo de informações uniforme para ser utilizado em todas as instâncias de GUACS. O modelo escolhido foi o UR (*Usage Record*) [UR, 2005], apresentado no Capítulo 3 (vide Tabela 3.1). Esse modelo agrega informações relacionadas à execução de aplicações e ao consumo de recursos, atendendo às necessidades deste trabalho.

A partir do modelo UR, várias informações gerenciais podem ser extraídas. Em relação às aplicações executadas, pode-se obter, por exemplo, o número de *jobs* executados em um determinado período de tempo (opcionalmente, com o estado final de computação desses *jobs*), a comparação do tempo de execução de vários *jobs* e o tempo de computação dos *jobs* pertencentes a um determinado usuário.

No que diz respeito a informações sobre recursos, é possível, por exemplo, determinar quais as estações que mais contribuíram com recursos para a grade, identificar estações que não estão contribuindo de forma produtiva (doando poucos recursos para a grade) e verificar o tráfego de rede comumente utilizado pela grade em determinadas horas do dia (permitindo reconhecer tendências de uso).

Por fim, referindo-se à associação das informações sobre as aplicações executadas com as informações sobre os recursos consumidos, pode-se identificar o impacto da execução

de uma dada aplicação sobre a infra-estrutura, observar em quais estações determinados *jobs* foram computados, e, ainda, a partir de quais escalonadores determinados *jobs* foram disparados para execução. Pode-se, também, comparar o consumo dos recursos (como carga de CPU, consumo de memória, espaço em disco e banda de rede) utilizados por determinados *jobs*.

Interface de Gerenciamento WSDM

O serviço de contabilização e caracterização organiza objetos de gerenciamento em uma árvore de tópicos, ou *propriedades*, por intermédio dos quais *publishers* atualizam informações e a aplicação de gerenciamento as consome. Os *publishers* atualizam informações junto ao serviço invocando requisições SETRESOURCEPROPERTIES. Já uma aplicação de gerenciamento pode: (a) solicitar o valor associado a uma propriedade (GETRESOURCEPROPERTY); (b) solicitar o valor associado a um conjunto filtrado de propriedades através de uma consulta XPath (QUERYRESOURCEPROPERTIES); e (c) (de)subscrever por uma ou mais propriedades (SUBSCRIBE/DESTROY), recebendo notificações quando as mesmas são atualizadas.

As propriedades que constituem o serviço são cinco: JOBSTARTED, JOBCOMPLETED, JOBABORTED, JOBFAILED e JOBHISTORY. A Tabela 4.1 sintetiza as operações admitidas e as informações fornecidas para cada uma delas. As quatro primeiras podem ter seus valores atualizados através de requisições SETRESOURCEPROPERTIES, além de poderem receber solicitações por subscrições. Nesse último caso, assumindo que essas propriedades tenham sido subscritas, a aplicação de gerenciamento receberá notificações, em tempo "quase real", sempre que uma aplicação concluir sua execução, de forma bem sucedida ou não. Por outro lado, a propriedade JOBHISTORY – que armazena múltiplas instâncias do registro UR (uma para cada *job* concluído com sucesso ou não) – só pode ser consultada através de requisições GETRESOURCEPROPERTY ou QUERYRESOURCEPROPERTIES.

As informações fornecidas pela propriedade JOBSTARTED indicam o estado inicial de execução de uma determinada aplicação, incluindo data e hora de início e estação onde foi escalonada. As informações oferecidas pelas propriedades JOBCOMPLETED, JOBABORTED e JOBFAILED são complementares às oferecidas por JOBSTARTED e revelam o estado final de execução dessa aplicação. Sempre que uma aplicação conclui sua execução é acrescentado à propriedade JOBHISTORY um registro UR, cujos campos são preenchidos com informações

oriundas da propriedade JOBSTARTED e uma das três propriedades: JOBCOMPLETED, JOBABORTED ou JOBFAILED.

Tabela 4.1. Propriedades fornecidas por GUACS.

Propriedade	Operações admitidas	Informações fornecidas[1]
JobStarted	SetResourceProperties, Subscribe, Destroy	JobId, StartTime, JobName, FromHost, ExecHost, UserName, ProjectName, Queue, GridId
JobCompleted, JobAborted, JobFailed	SetResourceProperties, Subscribe, Destroy	JobId, EndTime, Processors, NumNodes, CpuTime, WallTime, Memory, Disk, Network, Charge, Status
JobHistory[2]	GetResourceProperty, QueryResourceProperties	JobId, StartTime, EndTime, JobName, FromHost, ExecHost, UserName, ProjectName, Queue, GridId, Processors, NumNodes, CpuTime, WallTime, Memory, Disk, Network, Charge, Status

[1]As informações fornecidas estão em conformidade com o modelo *Usage Record*. [2]A propriedade JobHistory armazena múltiplas instâncias de UR.

4.3.2 Publishers

O funcionamento de um *publisher* pode ser organizado em três momentos. O primeiro consiste na observação da ocorrência de um evento pré-definido. O segundo prevê a normalização dos dados coletados, vinculados ao evento. Por fim, no terceiro momento uma mensagem de atualização é enviada ao serviço de contabilização e caracterização. A detecção da ocorrência de um evento e a normalização dos dados coletados são tarefas específicas a cada sistema de grade, estando fora do escopo do padrão WSDM. Já o envio da mensagem de atualização, através de uma requisição SETRESOURCEPROPERTIES, é realizado respeitando o mesmo.

Uma mensagem de atualização é sempre associada a um tópico (ex: JOBSTARTED ou JOBFAILED). Nos exemplos ilustrados na Figura 4.3, é possível observar o formato do conteúdo de requisições SETRESOURCEPROPERTIES invocadas para as propriedades JOBSTARTED (a) e JOBFAILED (b). Note que em cada operação *set* diversas informações, organizadas em um documento XML, são repassadas a GUACS.

```
<JobStarted>
   <JobId>1.1.1.nodo01.unisinos.br</JobId>
   <StartTime>2005-09-13 17:24:50</StartTime>
   <JobName>intracel_dynamics</JobName>
   <FromHost>nodo01.unisinos.br</FromHost>
   <ExecHost>nodo09.unisinos.br</ExecHost>
   <UserName>glauco_ludwig</UserName>
   <ProjectName>bio_paua</ProjectName>
   <Queue>nodo01</Queue>
   <GridId>glauco_ludwig@unisinos.br</GridId>
</JobStarted>
```

(a) JOBSTARTED

```
<JobFailed>
   <JobId>1.1.1.nodo01.unisinos.br</JobId>
   <EndTime>2005-09-13 17:30:50</EndTime>
   <Processors>2</Processors>
   <NumNodes>1</NumNodes>
   <CpuTime>PT15S</CpuTime>
   <WallTime>PT45M48S</WallTime>
   <Memory>1234</Memory>
   <Disk>560</Disk>
   <Network>1.000.000</Network>
   <Charge>300</Charge>
   <Status>failed</Status>
</JobFailed>
```

(b) JOBFAILED

Figura 4.3. Formato do conteúdo de uma requisição SETRESOURCEPROPERTIES.

4.3.3 Serviço de Autorização

O serviço de autorização provê um mecanismo de controle de acesso às informações mantidas pelo GUACS. Cada instância de GUACS tem associada a si uma instância do serviço de autorização. Essa descentralização permite que o administrador de cada instância do serviço de contabilização e caracterização defina, de forma autônoma, as políticas de divulgação de informações geradas em sua instituição. O projeto de um serviço de autorização independente de GUACS permite sua utilização e invocação, futuramente, por outros serviços de gerenciamento que venham a ser incorporados à arquitetura.

O serviço de autorização possui um repositório de políticas, estruturado de acordo com o modelo RBAC (*Role-Based Access Control*) [Sandhu, 2005]. RBAC foi escolhido devido à simplificação que proporciona no gerenciamento de autorizações. Conforme ilustra a Figura 4.4, políticas são vinculadas a papéis que, por sua vez, são associados a instituições.

Figura 4.4. Componenetes de uma política de divulgação de informações.

A Tabela 4.2 ilustra alguns exemplos de políticas que podem ser definidas em uma instância do serviço de autorização. Três papéis são definidos: *University*, *Enterprise* e *Default*. As políticas vinculadas a *University* especificam que o requisitante que se enquadrar nesse papel poderá acessar todos os valores associados a todas as propriedades fornecidas por GUACS. Já o papel *Enterprise* é mais restritivo. Ele permite que apenas a propriedade JOBHISTORY seja acessada. Além disso, os valores divulgados para cada registro se restringem a 9 de um total de 19 disponíveis (vide Tabela 4.1). Por fim, o papel *Default* não permite que

solicitações sejam respondidas pelo serviço. *University A* é associada ao papel *University*, enquanto *Company B* e *C* são associadas ao *Enterprise*.

Tabela 4.2. Exemplo de políticas definidas no serviço de autorização.

IP/Instituição	Papel	Políticas[1]
University A	University	JobStarted, JobCompleted, JobAborted, JobFailed, JobHistory JobId, StartTime, EndTime, JobName, FromHost, ExecHost, UserName, ProjectName, Queue, GridId, Processors, NumNodes, CpuTime, WallTime, Memory, Disk, Network, Charge, Status
Company B	Enterprise	JobHistory JobId, StartTime, EndTime, JobName, FromHost, GridId, CpuTime, Charge, Status
Company C	Enterprise	JobHistory JobId, StartTime, EndTime, JobName, FromHost, GridId, CpuTime, Charge, Status
Other	Default	None

[1]Informações que podem ser distribuídas.

Imaginando que o conjunto de políticas apresentado na tabela acima está sendo empregado por uma Universidade chamada *University D*, caso a instituição *University A* efetue uma requisição por informações históricas à *D*, todos os dados associados à propriedade JOBHISTORY serão divulgados. Já se essa mesma requisição for realizada a partir de *Company B* ou *Company C*, apenas um conjunto restrito de informações será compartilhado. Ainda, caso alguma das organizações pertencentes ao papel *Enterprise* solicite a subscrição por alguma propriedade, esse pedido não será autorizado pela *University D*.

4.3.4 Aplicação de Gerenciamento

Através de uma aplicação de gerenciamento, o administrador da grade deve configurar com que instâncias do serviço de contabilização e caracterização deseja interagir. Realizada essa configuração, duas funcionalidades básicas passam a ser oferecidas: subscrição por propriedades e recuperação de informações históricas junto a uma ou mais instâncias de GUACS. No primeiro caso, subscrições devem ser realizadas – através de requisições SUBSCRIBE – para que a aplicação de gerenciamento passe a receber notificações à medida em que eventos relevantes são observados na infra-estrutura de grade. Já no segundo caso, requisições GETRESOURCEPROPERTY ou QUERYRESOURCEPROPERTY são enviadas às instâncias de GUACS de interesse. As informações recebidas nas respostas podem ser

utilizadas para plotar gráficos variados, oferecendo diferentes visões sobre o uso global da infra-estrutura de grade.

A Figura 4.5 ilustra um exemplo de requisição de subscrição para a propriedade JOBFAILED, disparada a partir de uma aplicação de gerenciamento. Essa requisição, a exemplo das demais, segue rigorosamente o padrão WSDM. No exemplo é informado que as notificações deverão ser encaminhadas para o endereço http://www.unisinos.br na porta 8081 (local em que a aplicação de gerenciamento está preparada para receber as notificações). Já a Figura 4.6 apresenta parte de uma notificação recebida pela aplicação de gerenciamento em virtude de atualização no valor da propriedade JOBFAILED. Na notificação é possível observar o valor atualizado (representado por *NewValue*).

```
<wsnt:Subscribe>
   <wsnt:ConsumerReference>
      <wsa:Address>http://www.unisinos.br:8081</wsa:Address>
      <wsa:ReferenceProperties/>
   </wsnt:ConsumerReference>
   <wsnt:TopicExpressionDialect="http://docs.oasis-open.org/wsn/2004/
      06/TopicExpression/Simple">
      fs:JobFailed
   </wsnt:TopicExpression>
</wsnt:Subscribe>
```

Figura 4.5. Formato simplificado de uma subscrição para a propriedade JOBFAILED.

```
<wsn:Message>
   <wsrf:ResourcePropertyValueChangeNotification
    xmlns:wsrf="http://docs.oasis-open.org/wsrf/2004/06/wsrf-WS-
    ResourceProperties-1.2-draft-01.xsd">
      <wsrf:NewValue>
         <fs:JobFailed xmlns:wsrp="http://docs.oasis-
          open.org/wsrf/2004/06/wsrf-WS-ResourceProperties-1.2-draft-
          01.xsd" xmlns=http://schemas.xmlsoap.org/soap/envelope/
          xmlns:fs="http://ws.apache.org/resource/GUACS">
          <fil:JobId>1.1.1.nodo09.unisinos.br</fil:JobId>
          <fil:EndTime>2005-09-13T17:24:50Z</fil:EndTime>
          <fil:Processors>2</fil:Processors>
          <fil:NumNodes>1</fil:NumNodes>
          <fil:CPUTime>PT15S</fil:CPUTime>
          <fil:WallTime>PT45M48S</fil:WallTime>
          <fil:Memory>1234</fil:Memory>
          <fil:Disk>560</fil:Disk>
          <fil:Network>1.000.000</fil:Network>
          <fil:Charge>300</fil:Charge>
          <fil:Status>failed</fil: Status>
         </fs:JobFailed>
      </wsrf:NewValue>
   </wsrf:ResourcePropertyValueChangeNotification>
</wsn:Message>
```

Figura 4.6. Formato simplificado de uma notificação recebida pela aplicação de gerenciamento.

A Figura 4.7 exemplifica o formato do retorno de uma requisição do tipo GETRESOURCEPROPERTY. Percebe-se que essa requisição retorna uma coleção de registros UR associados à propriedade JOBHISTORY. Nota-se, também, que alguns campos estão com o valor *null* associado, podendo-se deduzir que restrições foram impostas pelo serviço de autorização presente na instituição para a qual a requisição foi efetuada.

```
<wsrf:GetResourcePropertyResponse xmlns:wsrf="http://docs.oasis-
 open.org/wsrf/2004/06/wsrf-WS-ResourceProperties-1.2-draft-01.xsd">
    <guacs:JobHistory xmlns:guacs="http://ws.apache.org/resource/GUACS">
       <guacs:JobHistory>
          ...
       </guacs:JobHistory>
       <guacs:JobHistory>
          <fil:JobId>1.1.2.nodo01.unisinos.br</fil:JobId>
          <fil:StartTime>2005-09-13 17:24:51</fil:StartTime>
          <fil:EndTime>2005-09-13 19:32:27</fil:EndTime>
          <fil:JobName>intracel_dynamics</fil:JobName>
          <fil:FromHost>nodo01.unisinos.br</fil:FromHost>
          <fil:ExecHost>nodo10.unisinos.br</fil:ExecHost>
          <fil:UserName>glauco_ludwig</fil:UserName>
          <fil:ProjectName>bio_paua</fil:ProjectName>
          <fil:Queue>nodo01</fil:Queue>
          <fil:GridId>glauco_ludwig@unisinos.br</fil:GridId>
          <fil:Processors>null</fil:Processors>
          <fil:NumNodes>null</fil:NumNodes>
          <fil:CPUTime>null</fil:CPUTime>
          <fil:WallTime>null</fil:WallTime>
          <fil:Memory>null</fil:Memory>
          <fil:Disk>null</fil:Disk>
          <fil:Network>null</fil:Network>
          <fil:Charge>null</fil:Charge>
          <fil:Status>completed</fil: Status>
      </guacs:JobHistory>
      <guacs:JobHistory>
          ...
      </guacs:JobHistory>
   </guacs:JobHistory>
</wsrf:GetResourcePropertyResponse>
```

Figura 4.7. Formato simplificado do retorno de uma requisição do tipo GETRESOURCEPROPERTY.

4.4 Implementação

Um protótipo da arquitetura proposta foi desenvolvido com o objetivo de avaliar a sua viabilidade técnica. Esta seção descreve aspectos relacionados com a implementação desse protótipo, incluindo informações sobre cada um dos componentes que compõem a arquitetura.

4.4.1 Serviço de Contabilização e Caracterização

O serviço GUACS foi implementado utilizando como base o *framework* Muse [Muse, 2005] – versão 1.0, da Apache Software Foundation, que se trata de uma implementação do

padrão WSDM. A partir da definição das propriedades que deveriam compor o serviço (JOBSTARTED, JOBCOMPLETED, JOBABORTED, JOBFAILED e JOBHISTORY), Muse foi estendido e o serviço, criado. Para armazenar os registros históricos relacionados com a propriedade JOBHISTORY, o banco de dados HSQLDB [HSQLDB, 2005] foi utilizado.

O ciclo de desenvolvimento de GUACS consituiu-se de quatro etapas. Na primeira delas foi criado o arquivo WSDL (*Web Services Description Language*), responsável pela descrição do serviço. Nesse arquivo foram informadas as propriedades e o tipo de requisições que seriam suportadas. Após, uma ferramenta para gerar código Java a partir do arquivo WSDL foi utilizada. Essa ferramenta é disponibilizada pela própria distribuição Muse. Em um terceiro momento, o código gerado na fase anterior foi instrumentado para implementar as funcionalidades oferecidas por GUACS (ex: a integração do serviço com a base de dados, já que MUSE não oferece um meio padrão para manter dados históricos). Por fim, o serviço foi compilado e disponibilizado através do servidor web Apache Tomcat [Tomcat, 2005].

4.4.2 Publishers

Os *publishers* foram desenvolvidos em *shell script*, capturando informações que são armazenadas em arquivos de *log*. Dois *publishers* foram desenvolvidos para duas diferentes tecnologias de grade: Globus e OurGrid. A escolha dessas tecnologias se deu em virtude da ampla aceitação que ambas vêm recebendo (a primeira delas no panorama internacional e, a segunda, pela comunidade de pesquisa nacional). A seguir, será apresentado como é feita a captura e a normalização de informações a partir do instante em que um evento de interesse é gerado pela tecnologia OurGrid. Para o desenvolvimento do *publisher* Globus, um processo semelhante ao que será ilustrado foi utilizado.

Em um sistema OurGrid, os arquivos de *log* encontram-se em todos os MyGrids (responsáveis pelo escalonamento das aplicações OurGrid) instalados pela grade – Sub-seção 2.2.3. Cada entrada no *log* (evento) é composta por quatro campos:

- *Timestamp*: informa a data e a hora em que um evento ocorreu;
- *Type*: indica o tipo de evento que foi gerado (i.e. informativo ou ocorrência de erro);
- *Method*: informa o método Java que originou o evento;
- *Message*: campo texto descrevendo o evento gerado pela infra-estrutura de grade.

A Figura 4.8 exemplifica o processo de captura de informações a partir do momento em que um evento de interesse é detectado pelo *publisher*. Nessa figura, é possível visualizar dois eventos de um *log* e as informações que são extraídas com o seu processamento: o evento

“1” informa a data e a hora na qual um *job* foi submetido (`2005-10-14 12:44:30`), a estação na qual ele foi designado a executar (`labo17.unisinos.br`) e a identificação desse *job* (`replica 1.1.1`). Já o evento de número “2” apresenta informações relacionadas ao término da execução do mesmo *job*, como a data e a hora em que foi finalizado e o estado final de sua computação (`Finished`).

```
  2005-10-14 12:44:30 INFO
1 org.ourgrid.mygrid.scheduler.Workqueue.schedule:105 ==>
  labo17.unisinos.br was assigned to execute replica 1.1.1
  2005-10-14 14:52:38 INFO
2 org.ourgrid.mygrid.scheduler.EBJobManager.replicaFinished:144 ==>
  Replica 1.1.1 finished on 'labo17.unisinos.br' with status 'Finished'
```

Figura 4.8. Estrutura dos arquivos de *log* gerados por um escalonador MyGrid e informações de interesse para o *publisher* OurGrid.

A Figura 4.9, por sua vez, ilustra o formato das requisições SETRESOURCEPROPERTIES invocadas para os dois eventos que foram capturados anteriormente (Figura 4.8). Em (a) percebe-se uma requisição *set* para a propriedade JOBSTARTED, originada no momento que o evento “1” é observado. Nota-se as atribuições dos valores extraídos aos campos *JobId*, *StartTime* e *ExecHost*. Outras informações como *FromHost* e *UserName* são obtidas usando variáveis de ambiente do próprio sistema operacional, já que essas informações não se fazem presentes nos *logs* do OurGrid. Já em (b) visualiza-se uma requisição *set* para a propriedade JOBCOMPLETED, onde nota-se as atribuições dos valores capturados aos campos *JobId* e *EndTime*. Percebe-se, também, a normalização da informação presente no campo *Status*. Apesar de o atributo extraído do *log* ser *finished*, esse foi normalizado para *completed*, estando assim de acordo com o sugerido pelo modelo UR.

```
<JobStarted>
   <JobId>1.1.1.nodo01.unisinos.br</JobId>
   <StartTime>2005-10-14 12:44:30</StartTime>
   <JobName>null</JobName>
   <FromHost>nodo01.unisinos.br</FromHost>
   <ExecHost>labo17.unisinos.br</ExecHost>
   <UserName>glauco_ludwig</UserName>
   <ProjectName>null</ProjectName>
   <Queue>null</Queue>
   <GridId>null</GridId>
</JobStarted>
```

(a) JOBSTARTED

```
<JobCompleted>
   <JobId>1.1.1.nodo01.unisinos.br</JobId>
   <EndTime>2005-10-14 14:52:38</EndTime>
   <Processors>null</Processors>
   <NumNodes>null</NumNodes>
   <CpuTime>null</CpuTime>
   <WallTime>null</WallTime>
   <Memory>null</Memory>
   <Disk>null</Disk>
   <Network>null</Network>
   <Charge>null</Charge>
   <Status>completed</Status>
</JobCompleted>
```

(b) JOBCOMPLETED

Figura 4.9. Requisições SETRESOURCEPROPERTIES geradas a partir de eventos capturados de *logs* OurGrid.

Observe na figura acima que as informações representadas por *null* se referem a registros relacionados com o consumo de recursos ou não oferecidas pela tecnologia de grade OurGrid. Ressalta-se, aqui, que por questões de limitação de escopo, agentes para a captura de informações sobre os recursos consumidos não foram implementados até o presente momento. No entanto, como pôde ser visto na Seções 4.2, GUACS foi projetado prevendo trabalhar com essa classe de informações e também com a associação dessas informações com as relacionadas à execução de aplicações.

4.4.3 Serviço de Autorização

Em relação ao serviço de autorização, destaca-se que o mesmo foi implementado em Java, sendo o repositório de políticas representado em arquivos XML. Um dos benefícios obtidos pela utilização desses arquivos é que podem ser lidos rapidamente e armazenados em estruturas de memória, minimizando o tempo de acesso às políticas. Além disso, deve-se considerar que XML tem se consolidado como o padrão para representação e intercâmbio de informações. A estrutura dos arquivos XML segue a especificação XACML [Godik et al., 2003], um padrão criado pelo OASIS que tem como objetivo auxiliar na especificação de políticas (com suporte a RBAC). Empregou-se, na implementação, a biblioteca XACML da Sun [SUNXACML, 2004].

O repositório de políticas está dividido em duas partes: (a) repositório de instituições e (b) repositório de papéis. O repositório de instituições tem como objetivo enquadrar os participantes em papéis, onde cada papel agrega as instituições que possuem o mesmo perfil de acesso às informações (por exemplo: as instituições *Company A* e *Company B* são associadas ao papel *Restricted*). Já no repositório de papéis são especificadas as políticas de acesso que cada papel possui junto ao serviço de autorização (por exemplo: o papel *Restricted* permite que apenas um conjunto específico de informações relacionadas à propriedade JOBFAILED seja divulgado e não permite que subscrições sejam realizadas).

4.4.4 Aplicação de Gerenciamento

A aplicação de gerenciamento, quarto componente da arquitetura, também foi desenvolvida em Java fazendo uso da biblioteca JFreeChart (para gerar gráficos) [JFreeChart, 2005]. Atualmente trabalha-se, também, no desenvolvimento de uma versão da aplicação para

operar de forma integrada à plataforma de gerenciamento HP OpenView [OpenView, 2005]. A Figura 4.10 ilustra a *Graphical User Interface* (GUI) da ferramenta.

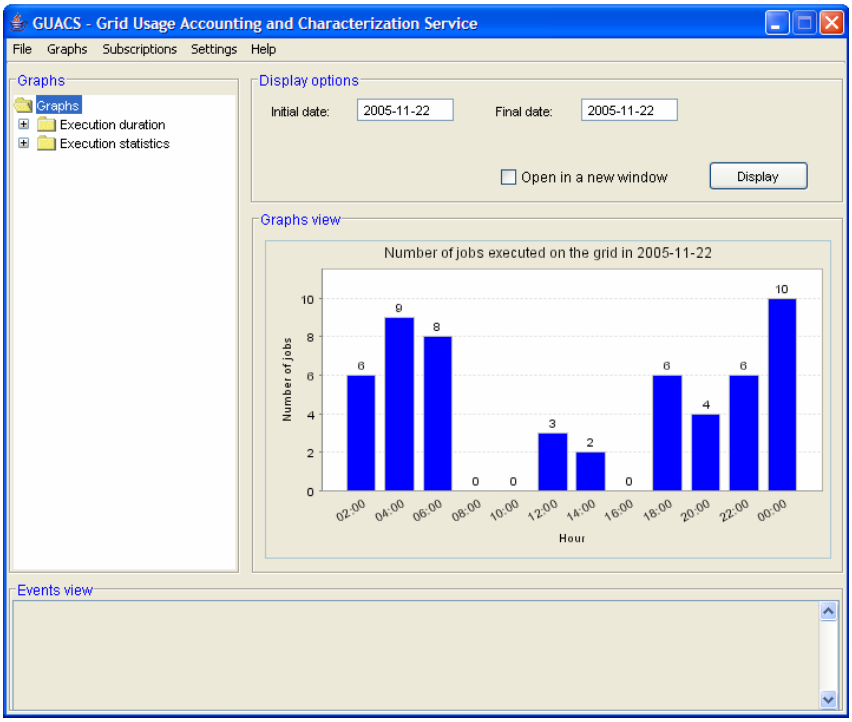


Figura 4.10. Interface gráfica da aplicação de gerenciamento.

As funcionalidades que se destacam na GUI são: (a) *Graphs*; (b) *Subscriptions*; e (c) *Settings*. Em *Graphs* (tanto na opção presente no menu principal quanto na árvore de opções disposta no lado esquerdo da ferramenta) é possível optar pela visualização de informações históricas. Após ter selecionado uma das informações disponíveis, o usuário escolhe no painel *Display options* opções mais detalhadas, como um intervalo de tempo (quando este for o caso). Fazendo uso do botão *Display*, o gráfico é gerado e apresentado no painel *Graphs View*, ou, ainda, aberto em uma nova janela (caso tenha sido feita a escolha *Open in a new window* no painel *Display options*). Essa opção pode ser extremamente útil quando se deseja gerar vários gráficos e compará-los lado a lado.

Já através da opção *Subscriptions*, é possível (a) subscrever por eventos, (b) destruir subscrições já realizadas e (c) visualizar as subscrições vigentes. A partir do momento em que uma subscrição é realizada com sucesso, as notificações recebidas passam a ser postadas para visualização no painel *Events view*. A Figura 4.11 ilustra a tela de subscrições. Nela, o usuário e/ou administrador da grade opta pelo GUACS (*service*) e tópico (*topic*) para o qual deseja realizar a subscrição.

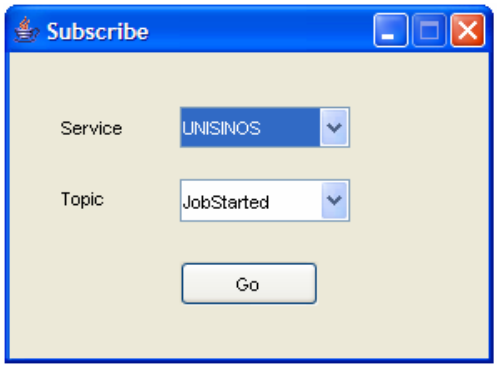


Figura 4.11. Pop-up para realizar subscrições.

Por fim, a opção *Settings* oferece ao usuário diferentes configurações como, por exemplo, o cadastro dos GUACS com os quais se deseja comunicar, bem como a porta nas quais os mesmos estão disponibilizados. Essa opção possibilita também a escolha das intâncias de GUACS que serão consultadas quando um gráfico for solicitado.

5 Avaliação Experimental

Este capítulo apresenta uma avaliação experimental realizada com a implementação da arquitetura. A arquitetura foi instanciada em um ambiente real de grade, com o objetivo de avaliar a sua capacidade em (a) prover uma visão global e uniforme do uso da infra-estrutura implantada e (b) em possibilitar a definição e a verificação de políticas de divulgação das informações coletadas. Na Seção 5.1 é descrita a instanciação da arquitetura, e na Seção 5.2, os resultados que foram obtidos.

5.1 Instanciação da arquitetura

Implantou-se uma infra-estrutura real de grade, como ilustra a Figura 5.1, constituída por três domínios administrativos: *Institution A*, *Institution B* e *Institution C*. As três instituições têm disponível, em seus domínios, escalonadores Globus (representados na figura por E_1) que compartilham uma estação presente em *Institution B*. Essa estação está apta a executar aplicações oriundas desses escalonadores. Outra tecnologia de grade presente na infra-estrutura é o OurGrid. Os escalonadores OurGrid (representados por E_2) formam uma rede *peer-to-peer* para o compartilhamento de recursos entre as três instituições. Foram utilizadas duas estações (recursos) para computar aplicações OurGrid em cada instituição.

Em relação à arquitetura de gerenciamento, uma instância do serviço de contabilização e caracterização foi instalada em cada domínio. Na figura, GUACS está representando simultaneamente o serviço de contabilização e caracterização, o(s) *publisher*(*s*) e o serviço de autorização. Todas as estações envolvidas no ambiente de avaliação (desde aquelas onde os componentes da arquitetura foram instanciados até aquelas aptas a executarem aplicações) tiveram seus relógios sincronizados com um servidor NTP (Network Time Protocol) [NTP, 2005].

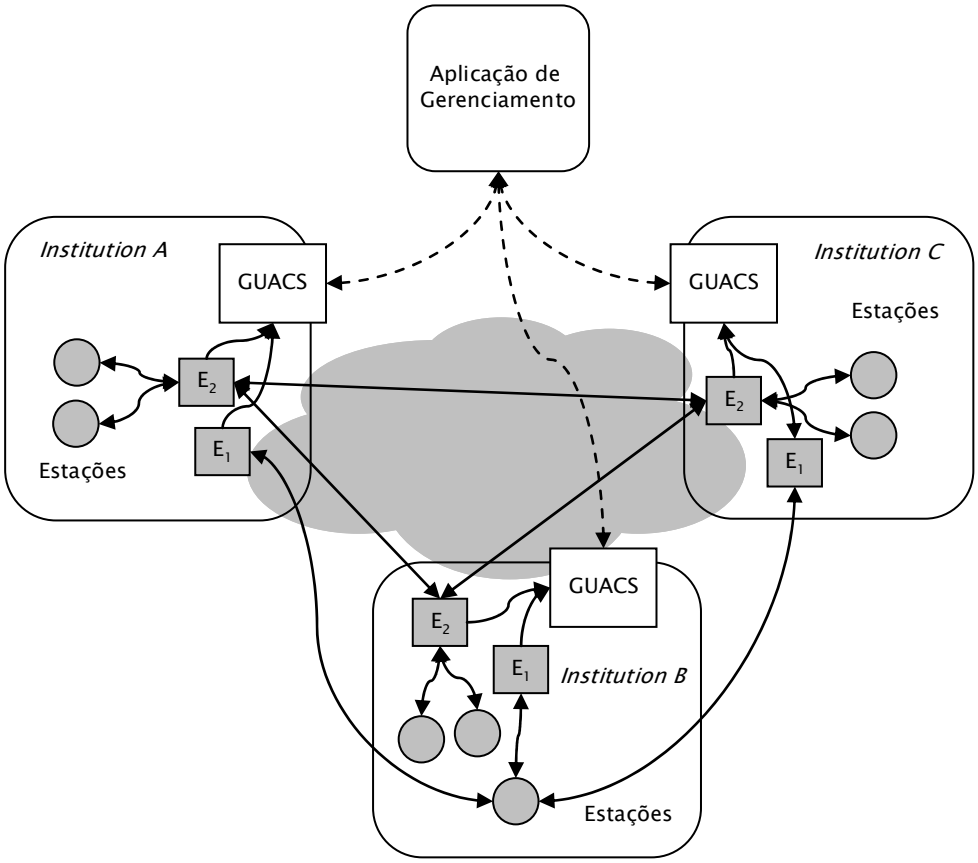


Figura 5.1. Infra-estrutura de grade utilizada para a avaliação da arquitetura.

O serviço de autorização instanciado em *Institution A* e em *Institution B* foi configurado de acordo com o conjunto de políticas definido na Tabela 5.1. Pelo fato dessas duas instituições serem parceiras, não há restrições entre elas para o acesso às informações. Ambas estão associadas ao papel *Unrestricted*. Por outro lado, *Institution C* está enquadrada no papel *Restricted*. As políticas definidas para esse papel permitem que apenas um conjunto específico de informações relacionadas à propriedade JOBHISTORY seja divulgado (9 registros em um total de 19 disponíveis) e não permitem que subscrições sejam realizadas.

O serviço de autorização presente em *Institution C*, por sua vez, foi configurado de acordo com o conjunto de políticas definido na Tabela 5.2. Percebe-se que não há restrições impostas para a divulgação de informações para a própria instituição. *Institution C* está associada ao papel *Unrestricted*. Já *Institution A* e *B*, associadas ao papel *Restricted*, mesmo tendo acesso a todas as propriedades fornecidas por GUACS (JOBSTARTED, JOBCOMPLETED, JOBABORTED, JOBFAILED e JOBHISTORY), poderão visualizar apenas um conjunto de 9 registros dos 19 disponíveis.

Tabela 5.1. Políticas definidas no serviço de autorização de *Institution A* e *Institution B.*

IP/Instituição	Papel	Políticas[1]
Institution A	Unrestricted	JobStarted, JobCompleted, JobAborted, JobFailed, JobHistory JobId, StartTime, EndTime, JobName, FromHost, ExecHost, UserName, ProjectName, Queue, GridId, Processors, NumNodes, CpuTime, WallTime, Memory, Disk, Network, Charge, Status
Institution B	Unrestricted	JobStarted, JobCompleted, JobAborted, JobFailed, JobHistory JobId, StartTime, EndTime, JobName, FromHost, ExecHost, UserName, ProjectName, Queue, GridId, Processors, NumNodes, CpuTime, WallTime, Memory, Disk, Network, Charge, Status
Institution C	Restricted	JobHistory JobId, StartTime, EndTime, JobName, FromHost, GridId, CpuTime, Charge, Status
Other	Default	None

[1]Informações que podem ser distribuídas.

Tabela 5.2. Políticas definidas no serviço de autorização de *Institution C.*

IP/Instituição	Papel	Políticas[1]
Institution C	Unrestricted	JobStarted, JobCompleted, JobAborted, JobFailed, JobHistory JobId, StartTime, EndTime, JobName, FromHost, ExecHost, UserName, ProjectName, Queue, GridId, Processors, NumNodes, CpuTime, WallTime, Memory, Disk, Network, Charge, Status
Institution A	Restricted	JobStarted, JobCompleted, JobAborted, JobFailed, JobHistory JobId, StartTime, EndTime, JobName, FromHost, GridId, CpuTime, Charge, Status
Institution B	Restricted	JobStarted, JobCompleted, JobAborted, JobFailed, JobHistory JobId, StartTime, EndTime, JobName, FromHost, GridId, CpuTime, Charge, Status
Other	Default	None

[1]Informações que podem ser distribuídas.

Para concretizar essa etapa da avaliação, uma aplicação que efetua computações sobre um modelo determinístico de dinâmica intracelular de um vírus foi utilizada [Srivastava, 2002]. Essa aplicação caracteriza-se por ser do tipo *bag-of-tasks* e foi ajustada para ser disparada tanto a partir dos escalonadores Globus quanto a partir dos escalonadores OurGrid.

A partir de uma aplicação de gerenciamento posicionada em *Institution A*, realizou-se subscrições – às propriedades JOBSTARTED, JOBCOMPLETED, JOBABORTED e JOBFAILED –

junto aos três serviços de contabilização e caracterização dispersos pelo ambiente. Como a aplicação utilizada na avaliação caracteriza-se por ser do tipo *bag-of-tasks* (composta por 16 mil *jobs*), para este experimento, apenas um pequeno conjunto desses *jobs* foi disparado (15 *jobs* por cada escalonador OurGrid e 5 *jobs* por cada escalonador Globus). O experimento demorou 15 horas e 12 minutos para ser finalizado.

5.2 Resultados obtidos

Durante o decorrer da computação dos 60 *jobs*, foi possível acompanhar o uso da infra-estrutura na medida em que notificações reportando o início e o término da execução desses *jobs* foram sendo recebidos e apresentados na aplicação de gerenciamento. Mesmo com duas tecnologias de grade envolvidas na infra-estrutura, GUACS comportou-se como esperado, sem fazer distinção entre os dois sistemas. Entre algumas das informações relevantes – além de poder acompanhar, por exemplo, para quais estações os *jobs* estavam sendo submetidos e o estado final de suas computações – identificou-se uma situação inesperada: uma das estações disponíveis no domínio *Institution B* estava falhando a computação de *jobs* com grande freqüência, atrasando o término da execução da aplicação. No momento em que o problema foi bem caracterizado (diversas ocorrências da notificação JOBFAILED) a estação foi excluída da infra-estrutura para inspeção.

A visão global do uso da grade se fez valer também para as consultas históricas que foram realizadas junto aos GUACS. A Figura 5.2 exibe um gráfico gerado pela aplicação de gerenciamento. Essa figura ilustra o número de *jobs* executados nas estações pertencentes ao domínio *Institution B*, onde é possível visualizar quantos desses *jobs* terminaram sua execução com sucesso e quantos falharam ou foram abortados.

Se analisado com o devido cuidado, esse gráfico revela a presença de uma estação com pouca contribuição para a grade. Observa-se que a estação 10.16.165.6 concluiu 4 *jobs* com o estado *Completed* e 5 com o estado *Failed*. Como mencionado anteriormente, essa estação foi retirada da infra-estrutura, já que a mesma estava degradando o desempenho da grade e da aplicação que estava sendo executada. Esse tipo de informação poderia ser repassado ao escalonador, que poderia guiar suas decisões levando esse *feedback* em consideração.

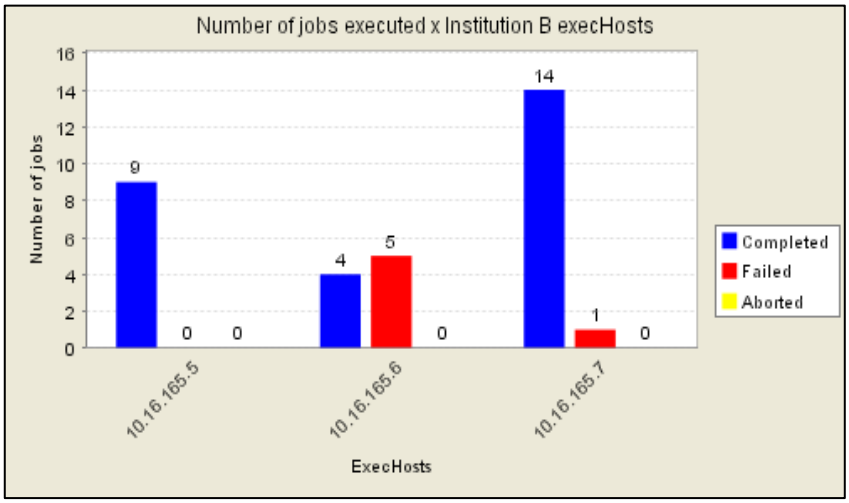


Figura 5.2. Número de *jobs* executados x *Institution B* ExecHost.

Na Figura 5.3 pode ser visto o número de *jobs* que executaram na grade no dia 23 de novembro de 2005. Pode-se perceber que foram computados 60 *jobs* ao longo desse dia. O tipo de resultado gerado por esse gráfico, além de contabilizar informações relacionadas às aplicações que executaram na grade, auxilia na identificação de padrões e tendências de uso da mesma. De posse de tais informações é possível, por exemplo, determinar com mais precisão necessidades de ampliação da infraestrutura.

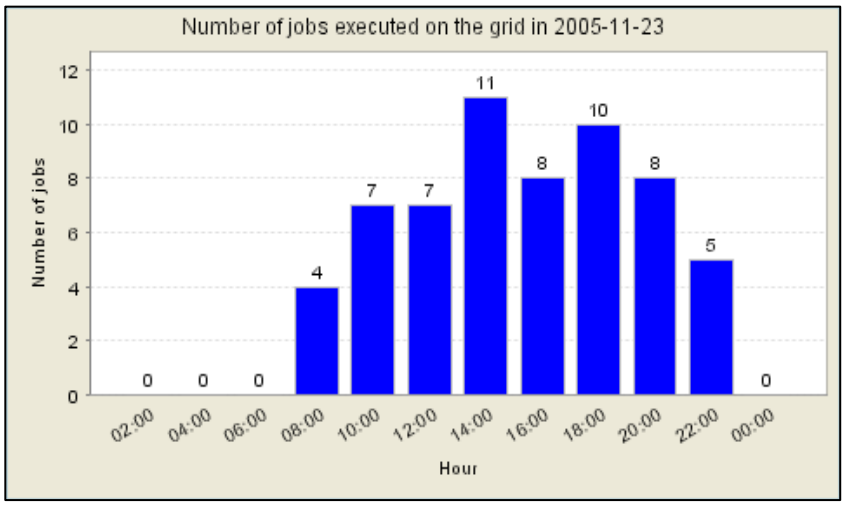


Figura 5.3. Número de *jobs* executados ao longo do dia 23 de novembro de 2005.

O experimento realizado permitiu observar, ainda, a arquitetura fazendo cumprir as políticas de divulgação definidas em cada um dos três domínios. Considerando que a aplicação de gerenciamento esteve posicionada no domínio *Institution A*, foi possível receber

em tempo “quase real” notificações geradas no próprio domínio *Institution A*, bem como nos domínios *Institution B* e *Institution C*. No entanto, as informações provindas de *Institution C* eram recebidas com um número menor de valores associados devido às restrições que foram impostas por essa instituição em relação ao domínio *Institution A*. O mesmo ocorreu com a recuperação de informações históricas (propriedade JOBHISTORY). As requisições enviadas ao domínio *Institution B* foram respondidas com registros UR completos, ao passo que as solicitações endereçadas à *Institution C* foram respondidas com registros incompletos.

Analisando os resultados que foram obtidos com a avaliação experimental, pôde-se perceber que mesmo com duas tecnologias de grade envolvidas na infra-estrutura, GUACS comportou-se como esperado, sem fazer distinção entre os dois sistemas. Foi possível acompanhar de forma global e uniforme o uso da grade tanto na medida em que os eventos iam ocorrendo quanto através de informações históricas. Pôde-se verificar, também, o funcionamento do serviço de autorização, fato que se comprova por *Institution A* ter recebido apenas um conjunto limitado de registros para as requisições que efetuou junto à *Institution C* (vide políticas definidas na Tabela 5.2). De maneira geral, conclui-se que ainda que incompleta (pela adoção de uso de políticas e pela ausência de agentes capazes de coletar informações sobre o consumo de recursos), obteve-se uma visão global e homogênea do uso da infra-estrutura.

6 Considerações Finais

A incorporação de mecanismos de gerenciamento de contabilização em ambientes de grades computacionais é de extrema relevância. Soluções com esse propósito podem auxiliar, por exemplo, na busca por uma compreensão mais precisa e detalhada sobre o uso da grade, encorajando seu emprego em ambientes mais sensíveis como o corporativo.

As soluções existentes para a gerência de grades, contudo, são limitadas (a) por se restringirem a monitorar apenas o estado dos recursos disponíveis no ambiente (como carga de CPU e consumo de memória), ignorando o olhar sob a perspectiva das aplicações que são executadas; (b) por não oferecem suporte à coleta, ao processamento e à consolidação de informações que são geradas por diferentes tecnologias de grade; e (c) por não permitirem a especificação de políticas de divulgação das informações coletadas.

Esta dissertação apresentou uma abordagem, acompanhada de uma arquitetura, para a contabilização e a caracterização de uso de grades computacionais que supre as três limitações supracitadas. Para demonstrar o conceito e a viabilidade técnica da proposta, um protótipo da arquitetura foi desenvolvido e instanciado em um ambiente real de grade. Os resultados obtidos demonstram que a arquitetura proposta pode ser aplicada em situações reais, fato comprovado pela implementação de um protótipo capaz de monitorar a execução de uma aplicação real, do seu lançamento ao seu término. Diversas informações gerenciais puderam ser obtidas, como, por exemplo, a comparação do tempo de execução de vários *jobs*, o tempo de computação dos *jobs* pertencentes a um determinado usuário e a visualização das estações em que determinados *jobs* foram computados.

A arquitetura foi desenvolvida com base no padrão WSDM, padronizado pelo OASIS em março de 2005. Aqui, o objetivo foi projetar e implementar uma arquitetura de gerenciamento que estivesse em consonância com as tecnologias atuais de grades, as quais têm procurado organizar seus componentes na forma de serviços *web*. Até onde se sabe, este é o primeiro trabalho a propor uma arquitetura para gerenciar grades computacionais através do *framework* WSDM.

A atual implementação da arquitetura é limitada por não comtemplar a obtenção de informações estatísticas durante o decorrer da execução de um dado *job*, ou seja, os dados

atualmente contabilizados e caracterizados são obtidos apenas no início e no término da computação desse *job*. Assim, torna-se impossível caracterizar um comportamento anormal ou suspeito de uma dada aplicação (como no caso de um ataque de negação de serviço) durante o decorrer de sua computação. Como trabalho futuro, pretende-se estender a arquitetura para suportar esse tipo de situação. Ainda como trabalho futuro, deseja-se concluir a implementação de agentes de software capazes de coletar informações sobre o consumo de recursos, então, será possível alimentar GUACS com essa classe de informações e também associá-las com as informações sobre a execução de aplicações. Além disso, pretende-se incorporar a arquitetura à infra-estrutura de grade Pauá, idealizada pela HP Computadores do Brasil e que tem como objetivo avançar o uso e a pesquisa em grades no Brasil.

Referências Bibliográficas

[Andrade et al., 2003] Andrade, N. et al. (2003) "OurGrid: An Approach to Easily Assemble Grids with Equitable Resource Sharing", In: 9th International Workshop on International Job Scheduling Strategies for Parallel Processing, p. 61-86.

[Baker, 2000] Baker, M., Buyya, R., Laforenza, D. (2000) "The Grid: international efforts in global computing.", In: International Conference on Advances in Infrastructure for Eletronic Business, Science, and Education on the Internet.

[Baker, 2003] Baker, M. and Smith, G. (2003) "GridRM: An Extensible Resource Monitoring System", In: IEEE International Conference on Cluster Computing, p. 207-215.

[Berman, 2005] Berman, F., Gagliardi, F., Kesselman, C. and Linesch, M. (2005) "What will Grids look like in Five Years?", In: 6th IEEE/ACM International Workshop on Grid Computing.

[Booth et al., 2005] Booth et al. (2005) Web Services Description Language (WSDL) Version 2.0 Part 0. Working Draft. http://www.w3.org/TR/2005/WD-wsdl20-primer-20050803.

[Catania et al., 2003] Catania, N. et al. (2003) Web Service Management Framework (WSMF) Version 2.0. Committee Specification. http://devresource.hp.com/drc/specifications/wsmf/WSMF-Overview.jsp.

[Cerami, 2002] Cerami, E. (2002) "Web Services Essentials - Distributed Applications with XML-RPC, SOAP, UDDI & WSDL". O´Reilly.

[Chung, 2003] Chung, J. (2003) “Web Services Computing: Advancing Software Interoperability.” Published by the IEEE Computer Society, vol. 7, n. 6, 12-15.

[Cirne, 2003a] Cirne, W. (2003a) “Grids Computacionais: Arquiteturas, Tecnologias e Aplicações”, In: Anais da Terceira Escola Regional de Alto Desempenho, p. 103-134.

[Cirne, 2003b] Cirne, W. et al. (2003b) “Running Bag-of-Tasks Applications on Computational Grids: The MyGrid Approach”, In: Proceedings of the International Conference on Parallel Processing.

[Cirne et al., 2004] Cirne, W. et al. (2004) “Scheduling in Bag-of-Tasks Grids: The Pauá Case”, In: 16th Symposium on Computer Architecture and High Performance Computing.

[Clement et al., 2004] Clement, L. et al. (2004) Universal Description, Discovery and Integration (UDDI). Committee Draft. http://www.oasis-open.org/committees/uddi-spec/doc/spec/v3/uddi-v3.0.2-20041019.htm.

[Condor, 2005] Condor Project Home Page (2005) http://www.cs.wisc.edu/condor.

[Dinda et al., 2001] Dinda, P. et al. (2001) “The Architecture of the Remos System”, In: 10th IEEE International Symposium on High Performance Distributed Computing, p. 383-394.

[Distributed.net, 2005] Distributed.net Project Home Page (2005) http://distributed.net.

[Entropia, 2005] Entropia Project Home Page (2005) http://www.entropia.com.

[Ferris, 2003] Ferris, C. and Farrel, J. (2003) “What Are Web Services?”, Communication of the ACM, vol. 46, n. 6, p. 31.

[Foster, 1998] Foster, I., Kesselman, K. (1998) “The Grid: Blueprint for a New Computing Infrastructure”, Morgan Kaufmann Publishers, 1st edition.

[Foster et al., 2001] Foster, I. et al. (2001) “The Anatomy of the Grid: Enabling Scalable Virtual Organizations”, In: The International Journal of High Performance Computing Applications, vol. 15, n. 3, p. 200-222.

[Foster et al., 2002] Foster, I. et al. (2002) “The Physiology of the grid: An Open Grid Services Architecture for Distributed Systems Integration”, In: Global Grid Forum.

[Foster et al., 2005] Foster, I. et al. (2005) “Modeling and managing state in distributed systems: the role of OGSI and WSRF”, In: Proceedings of the IEEE, vol. 93, n. 3, p. 604-612.

[Geller et al., 2004] Geller, A. et al. (2004) Web Service for Management (WS-Management) Version for Public Comment. Committee Specification. http://msdn.microsoft.com/library/en-us/dnglobspec/html/ws-management.pdf.

[GGF, 2005] Global Grid Forum (GGF) Project Home Page (2005) http://www.gridforum.org.

[Globus, 2005] Globus Toolkit Home Page (2005) http://www.globus.org.

[Godik et al., 2003] Godik, S. et a l. (2003) eXtensible Access Control Markup Language (XACML). Version 1.1. Committee Specification. http://www.oasis-open.org/committees/xacml/repository/cs-xacml-specification-1.1.pdf.

[Graham et al., 2005] Graham, S. et al. (2005) Web Services Notification (WSN). Version 1.3. Committee Specification. http://www.oasis-open.org/committees/documents.php?wg_abbrev=wsn.

[GSI, 2005] GSI Project Home Page (2005) http://www.globus.org/security.

[Gudgin et al., 2003] Gudgin, M. et al. (2003) Simple Object Access Protocol (SOAP). Version 1.2 Part 1: Messaging Protocol. Committee Specification. http://www.w3.org/TR/2003/REC-soap12-part1-20030624/.

[Hamilton, 2004] Hamilton, D. (2004) http://www.microsoft.com/presspass/features/2004/oct04/10-08WSManagement.mspx.

[HSQLDB, 2005] HSQLDB Project Home Page (2005) http://hsqldb.org.

[Tivoli, 2005] IBM Tivoli Home Page (2005) http://www-306.ibm.com/software/tivoli.

[JFreeChart, 2005] JFreeChart Project Home Page (2005) http://www.jfree.org/jfreechart/index.php.

[Krauter et al., 2002] Krauter, K. et al., (2002) "A taxonomy and survey of grid resource management systems for distributed computing", In: Software: Practice and Expirience, vol. 32, n. 2, p. 135-164.

[Kumar, 2004] Kumar, P. (2004) http://www.pankaj-k.net/archives/2004/10/wsdm_versus_wsm.html.

[Lee, 2003] Lee, D., Dongarra, J. and Ramakrishna, R. (2003) "VisPerf: Monitoring Tool for Grid Computing", In: International Conference on Computational Science. Lecture Notes in Computer Science, vol. 2659, p. 233-243.

[Legion, 2005] Legion Project Home Page (2005) http://legion.virginia.edu.

[Ludwig, 2005a] Ludwig, G., Gaspary, L. Cavalheiro, G. (2005a) "MUAT: Um Ambiente para Contabilização e Caracterização de Uso de Grades Computacionais", In: 10th Workshop de Gerência e Operação de Redes e Serviços.

[Ludwig, 2005b] Ludwig, G., Gaspary, L. Cavalheiro, G. (2005b) "MUAT: An Environment for Accounting and Characterization of the Use of Computational Grids", In: 4thLatin American Network Operations and Management Symposium.

[Massie, 2004] Massie, M. L.; Chun, B. N.; Culler, D. E. (2004) "The Ganglia Distributed Monitoring System: design, implementation, and experience. In: Parallel Computing, vol. 30, n. 7, p. 817-840.

[MRTG, 2005] MRTG Project Home Page (2005) http://mrtg.hdl.com/mrtg.html.

[MUSE, 2005] MUSE Project Home Page (2005) http://ws.apache.org/ws-fx/muse.

[Nemeth, 2002] Nemeth, Z., Sunderam, V. (2002) "A formal framework for defining grid systems", In: 2th IEEE/ACM International Symposium on Cluster Computing and the Grid, p. 202-211.

[Netsolve, 2005] Netsolve Project Home Page (2005) http://icl.cs.utk.edu/netsolve.

[Newman, 2003] Newman, H. B. et al. (2003) "MonALISA: a distributed monitoring service architecture", In: Computing in High Energy and Nuclear Physics.

[NTP, 2005] Network Time Protocol (NTP) Project Home Page (2005) http://www.ntp.org.

[NWS, 2005] Network Weather Service (NWS) Project Home Page (2005) http://nws.cs.ucsb.edu.

[OASIS, 2005] OASIS Home Page (2005) http://www.oasis-open.org.

[OpenView, 2005] HP OpenView Home Page (2005) http://www.managementsoftware.hp.com.

[OurGrid, 2005] OurGrid Project Home Page (2005) http://www.ourgrid.org.

[RUS, 2005] Resource Usage Service (RUS) Project Home Page (2005) https://forge.gridforum.org/projects/rus-wg.

[Sandhu, 2005] Sandhu, R. et al. (2005) Proposed NIST Standard for Role-Based Access Control. http://csrc.nist.gov/rbac/rbacSTD-ACM.pdf.

[Sedukhin et al., 2005] Sedukhin et al. (2005) Management of Web Services (WSDM-MOWS) Version 1.0. OASIS Standard. http://docs.oasis-open.org/wsdm/2004/12/wsdm-mows-1.0.pdf.

[Sotomaior, 2005] Sotomaior, B. (2005) "The Globus Toolkit 4 Programmer's Tutorial". http://gdp.globus.org/gt4-tutorial.

[Srivastava, 2002] Srivastava, R.,Yin, L. and Yin, J. (2002) “Stochastic vs. Deterministic Modeling of Intracellular Viral Kinetics” Theory Biology, vol. 218, p. 309-321.

[SUNXACML, 2004] Sun’s XACML (2004) Implementation Programmer’s Guide. http://sunxacml.sourceforge.net/guide.html.

[Talia, 2002] Talia, D. (2002) “The Open Grid Services Architecture: Where the Grid Meets the Web”, In: Internet Computing, IEEE, vol. 6, n. 6, p. 67-71.

[Tierney, 2002] Tierney, B. et al. (2002) “A Grid Monitoring Architecture”, Technical Report, Global Grid Forum Performance Working Group.

[UR, 2005] Usage Record (UR) Project Home Page (2005) https://forge.gridforum.org/projects/ur-wg.

[Vambenepe et al., 2005] Vambenepe et al. (2005) Management Using Web Services (WSDM-MUWS) Version 1.0. OASIS Standard. http://docs.oasis-open.org/wsdm/2004/12/wsdm-muws-part1-1.0.pdf.

[Vargas, 2005] Vargas, P., Barreto, M. (2005) “Gerência de Recursos em Ambientes de Grade”, Minicurso, In: 5th Escola Regional de Alto Desempenho (ERAD), p.3-32.

[Vinoski, 2005] Vinoski, S. (2005) “Web Service References” In: IEEE Internet Computing, vol. 9, n. 3, p. 90-93.

Livros Grátis

(http://www.livrosgratis.com.br)

Milhares de Livros para Download:

Baixar livros de Administração
Baixar livros de Agronomia
Baixar livros de Arquitetura
Baixar livros de Artes
Baixar livros de Astronomia
Baixar livros de Biologia Geral
Baixar livros de Ciência da Computação
Baixar livros de Ciência da Informação
Baixar livros de Ciência Política
Baixar livros de Ciências da Saúde
Baixar livros de Comunicação
Baixar livros do Conselho Nacional de Educação - CNE
Baixar livros de Defesa civil
Baixar livros de Direito
Baixar livros de Direitos humanos
Baixar livros de Economia
Baixar livros de Economia Doméstica
Baixar livros de Educação
Baixar livros de Educação - Trânsito
Baixar livros de Educação Física
Baixar livros de Engenharia Aeroespacial
Baixar livros de Farmácia
Baixar livros de Filosofia
Baixar livros de Física
Baixar livros de Geociências
Baixar livros de Geografia
Baixar livros de História
Baixar livros de Línguas