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

Glauco Antonio Ludwig[1], Luciano Paschoal Gaspary[2], Gerson Geraldo Homrich Cavalheiro[3], Walfredo Cirne[4]

[1]Programa Interdisciplinar de Pós-Graduação em Computação Aplicada
Universidade do Vale do Rio dos Sinos (UNISINOS)
glaucol@unisinos.br

[2]Instituto de Informática
Universidade Federal do Rio Grande do Sul (UFRGS)
paschoal@inf.ufrgs.br

[3]Instituto de Física e Matemática
Universidade Federal de Pelotas (UFPEL)
gerson.cavalheiro@ufpel.tche.br

[4]Departamento de Sistemas e Computação
Universidade Federal de Campina Grande (UFCG)
walfredo@dsc.ufcg.edu.br

Abstract. Current grid computing accounting and characterization solutions face limitations since they (a) do not provide a global, uniform view of the use of infrastructures comprised of distinct grid systems, (b) do not allow the specification of policies to distribute the collected information, and (c) do not generate statistics about the applications that are executed on the grid. To fullfil this gap, this paper proposes an approach based on policies to account and characterize usage of grid computing, even when such grids are formed by heterogeneous systems. The approach is materialized through an architecture based on the Web Services Distributed Management standard. The paper presents this architecture and results obtained with its instantiation in a real setup.

Resumo. As soluções existentes para contabilizar e caracterizar o uso de grades computacionais são limitadas (a) por não oferecerem uma visão global e uniforme do uso de infra-estruturas compostas por sistemas de grade heterogêneos, (b) por não permitirem a especificação de políticas de divulgação das informações coletadas e (c) por não gerarem estatísticas sobre as aplicações que são executadas. Para suprir essa lacuna, este artigo propõe uma abordagem baseada em políticas para contabilização e caracterização de uso de grades computacionais compostas por sistemas heterogêneos. A abordagem proposta se materializa por meio de uma arquitetura baseada no padrão Web Services Distributed Management. O artigo apresenta essa arquitetura, bem como resultados obtidos com a sua instanciação em um ambiente real.

* Este trabalho foi desenvolvido em colaboração com a HP Brasil P&D.

1. Introdução

O interesse no uso da tecnologia de grades computacionais vem se expandindo de forma gradual e consistente nos últimos anos. Com o amadurecimento das soluções oferecidas – nos próximos três a cinco anos – a expectativa é que 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 de grade.

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 quais usuários contribuem com mais recursos e quais fazem mais uso do poder computacional;
- 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 e a alocação correta de aplicações, negando ou permitindo que usuários executem novas aplicações baseando-se em seu histórico de utilização (usuários que tenham excedido uma cota de uso não deveriam ter acesso aos recursos da grade);
- identificar estações que fazem parte da grade e que não estão contribuindo de forma produtiva (uma máquina recebe tarefas para computar e acaba falhando na execução dessas freqüentemente);
- acompanhar a evolução do seu uso (permitindo reconhecer padrões e tendências).

Diversas soluções para contabilização e caracterização de uso de grades computacionais foram propostas nos últimos anos. Entre elas, destacam-se as apresentadas em [Dinda 2001], [Tierney 2002], [Baker 2003], [Lee 2003], [Newman 2003] e [Massie 2004]. Todavia, essas soluções pecam ao não serem capazes de coletar, processar e consolidar informações geradas por infra-estruturas de grade construídas com sistemas distintos, como Globus [Globus 2005], Condor [Condor 2005] e OurGrid [OurGrid 2005]. Devido a essa limitação, instituições[1] que cooperam utilizando tecnologias de grade distintas – 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 de grade.

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

Soma-se às duas limitações recém comentadas uma terceira não menos relevante: as soluções se restringem a monitorar o estado dos recursos disponíveis na infra-estrutura, como consumo de memória e espaço de armazenamento em disco, omitindo dados estatísticos e históricos sobre a execução de aplicações. Com as soluções atuais é possível visualizar, por exemplo, que uma determinada estação esteve com sua carga de CPU extremamente elevada nas últimas doze horas. Contudo, não é possível identificar a razão pela qual isso está ocorrendo. Pelo fato de não abordarem informações sobre a computação de aplicações e tampouco relacionarem essas informações com os recursos consumidos, as abordagens atuais não oferecem uma compreensão precisa e detalhada do verdadeiro uso da grade.

Para suprir as deficiências enumeradas anteriormente, este artigo propõe uma abordagem, acompanhada de uma arquitetura, para a contabilização e a caracterização de uso de grades computacionais que: (a) é capaz de oferecer uma visão global e uniforme do uso de infra-estruturas compostas por sistemas distintos de grade; (b) permite a especificação de políticas de divulgação das informações coletadas; e (c) suporta a geração de informações estatísticas não apenas sobre o uso dos recursos que compõem a grade, mas também sobre as aplicações que são executadas, sendo possível associar a computação dessas aplicações com os recursos consumidos. 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 padrão WSDM (*Web Services Distributed Management*) [Vambenepe et al. 2005; Sedukhin et al. 2005], padronizado pelo OASIS em março de 2005. É importante destacar que, até onde sabemos, este é o primeiro trabalho a propor um componente para gerenciamento de grades através do *framework* WSDM. O artigo apresenta essa arquitetura, bem como resultados obtidos com a sua instanciação em um ambiente real.

O restante do artigo está organizado da seguinte forma: a Seção 2 descreve trabalhos relacionados. Na Seção 3 apresenta-se uma visão conceitual da arquitetura proposta, e na Seção 4 são descritos os componentes que compõem essa arquitetura. Como prova de conceito, a Seção 5 aborda o protótipo desenvolvido. Na Seção 6 é apresentada uma avaliação experimental da arquitetura. O artigo é encerrado na Seção 7 com consideraçõs finais e perspectivas de trabalhos futuros.

2. Trabalhos Relacionados

Sistemas como Remos [Dinda 2001], visPerf [Lee 2003], GridRM [Baker 2003], MonALISA [Newman 2003] e Ganglia [Massie 2004] não são capazes de operar sobre infra-estruturas heterogêneas, que compartilham recursos utilizando diferentes tecnologias de grade. Como cada tecnologia tende a empregar uma forma própria para representar indicadores sobre as aplicações executadas e os recursos consumidos, esses sistemas não estão preparados para coletar esses indicadores e normalizá-los para um modelo de informações uniforme que, em última análise, ofereceria uma visão de uso global 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.

MonALISA [Newman 2003] e RUS (*Resource Usage Service*) [RUS 2005] são as únicas abordagens que, de alguma forma, se preocupam em gerar informações relacionadas com a execução de aplicações na grade. No entanto, MonALISA falha por não vincular as informações de aplicações com os recursos consumidos, o que pode impedir uma compreensão mais profunda sobre o uso da infra-estrutura. RUS, 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, RUS não se preocupa com a forma como esses dados devem ser coletados, processados, consolidados e apresentados aos administradores da grade.

Em trabalho anterior [Ludwig et al. 2006], apresentamos um primeiro esboço de arquitetura para resolver os problemas supramencionados. Este artigo o estende incluindo mais informações sobre a implementação realizada e os experimentos conduzidos.

3. Visão Conceitual da Arquitetura

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 denominado GUACS (*Grid Usage Accounting and Characterization Service*), (b) *publishers*, (c) serviço de autorização e (d) aplicação de gerenciamento. A Figura 1 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 (E1 e E2).

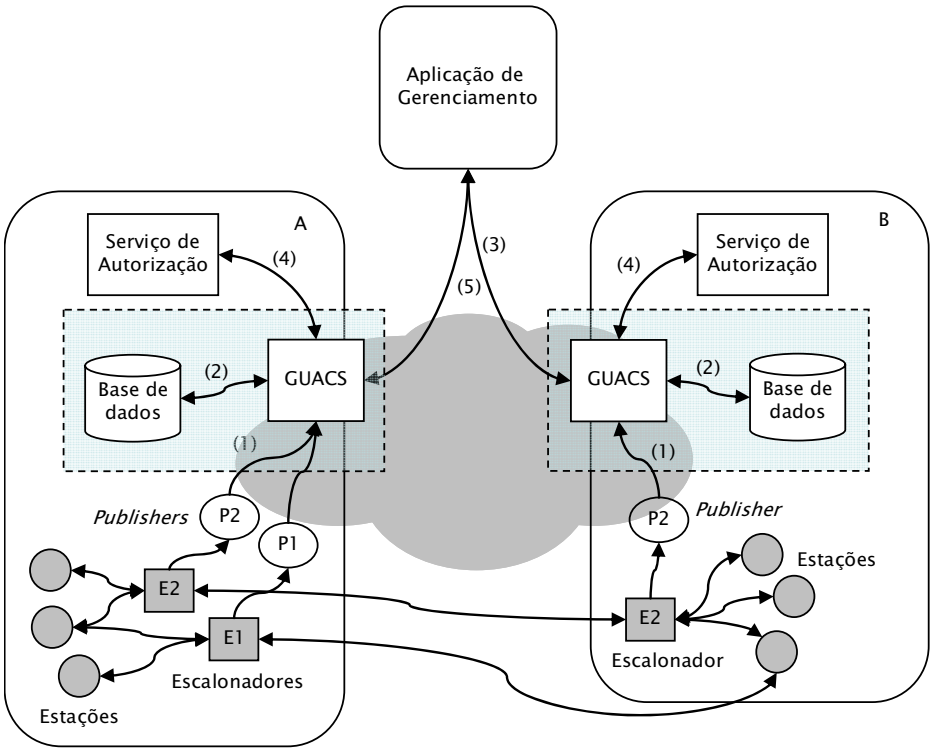


Figura 1. Visão geral da arquitetura e sua instanciação em grade envolvendo dois domínios administrativos.

O serviço de contabilização e caracterização 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 1). 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 notificar sobre eventos ocorridos na grade. Essas requisições, originadas a partir de uma aplicação de gerenciamento (fluxo 3 na Figura 1), 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 1, suponha que o cenário é composto por Globus (representado pelo escalonador E1) e OurGrid (escalonadores E2). Nesse caso, *publishers* específicos para Globus e OurGrid, representados na figura por P1 e P2, 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 1), 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 (a) reportando que o pedido foi negado ou (b) com informações completas ou parciais (5).

A aplicação de gerenciamento constitui o quarto, e último, componente da arquitetura. A partir dela, o administrador pode realizar subscrições em diversos GUACS para ser notificado sobre eventos ocorridos na infra-estrutura de grade. Adicionalmente, pode solicitar informações históricas de uma ou mais instâncias de GUACS e consolidá-las, visando a caracterizar e contabilizar 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. Componentes da Arquitetura

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.

4.1. 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: modelo de informação para armazenar as informações sobre o uso da grade e interface WSDM para comunicação com *publishers* e aplicação de gerenciamento.

4.1.1. Modelo de Informação

Como mencionado na Seção 3, 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], proposto pelo Global Grid Forum [GGF 2005]. Esse modelo é resultado de um esforço para definir um formato de registro que acomode indicadores empregados por diferentes tecnologias de grade existentes. A Tabela 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*), atendendo às necessidades do trabalho.

Tabela 1. O modelo de informação 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*[2] 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*

4.1.2. 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 seis: JOBSTARTED, JOBCURRENTSTATUS, JOBCOMPLETED, JOBABORTED, JOBFAILED e JOBHISTORY. A Tabela 2 sintetiza as operações admitidas e as informações fornecidas para cada uma delas. As cinco primeiras podem ter seus valores atualizados através de requisições SETRESOURCEPROPERTIES, além de poderem receber solicitações por subscrições. Nesse

[2] A palavra *job* refere-se, neste artigo, a uma unidade de execução (indivisível) manipulada pelo escalonador.

último caso, assumindo que essas propriedades tenham sido subscritas, a aplicação de gerenciamento receberá notificações, em tempo "quase real", sempre que um *publisher* observar a ocorrência de um evento. 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.

Tabela 2. Propriedades fornecidas por GUACS.

Propriedade	Operações admitidas	Informações fornecidas[1]
JobStarted	SetResourceProperties, Subscribe, Destroy	JobId, StartTime, JobName, FromHost, ExecHost, UserName, ProjectName, Queue, GridId
JobCurrentStatus	SetResourceProperties, Subscribe, Destroy	JobId, Processors, NumNodes, CpuTime, WallTime, Memory, Disk, Network, Charge
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.

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 JOBCURRENTSTATUS, JOBCOMPLETED, JOBABORTED e JOBFAILED são complementares às oferecidas por JOBSTARTED. Enquanto JOBCURRENTSTATUS informa o estado atual de execução de uma dada aplicação, as propriedades JOBCOMPLETED, JOBABORTED e JOBFAILED 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.

4.2. *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.

```
<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 2. Formato do conteúdo de uma requisição SETRESOURCEPROPERTIES.

Uma mensagem de atualização é sempre associada a um tópico (ex: JOBSTARTED ou JOBFAILED). Nos exemplos ilustrados na Figura 2, 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.

4.3. 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 seu *site*. 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 3, políticas são vinculadas a papéis que, por sua vez, são associados a instituições.

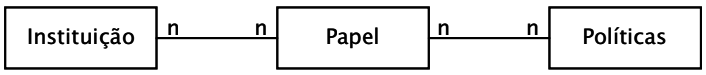


Figura 3. Componentes de uma política de divulgação de informações.

A Tabela 3 ilustra alguns exemplos de políticas que poderiam 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 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 3. Exemplos de políticas definidas no serviço de autorização.

IP/Instituição	Papel	Políticas[1]
University A	University	JobStarted, JobCurrentStatus, 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

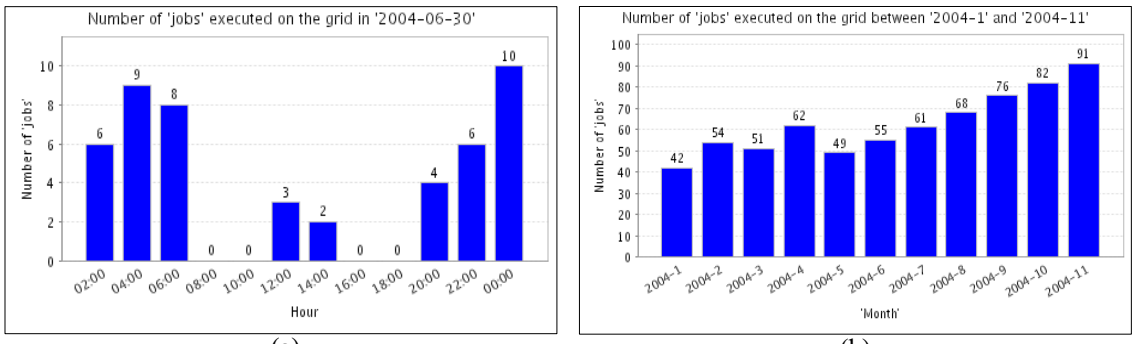
[1]Informações que podem ser distribuídas.

4.4. 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 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 ilustra o número de *jobs* que executaram na grade em um determinado período de tempo. Em (a) é representado o número de *jobs* que finalizaram sua execução em determinadas horas do dia 30 de junho. É possível perceber que esses se concentram em horário noturno e ao meio dia, podendo-se atribuir tal fenômeno ao fato da adoção do uso de políticas pela grade, as quais impedem seu funcionamento em horários em que há uma maior demanda por processamento local. Em (b) está representado o número de *jobs* que finalizaram sua execução entre os meses de janeiro e novembro. Observa-se que a partir do mês de agosto a grade passou a ser usada para executar um número cada vez maior de *jobs*. 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 pode-se, por exemplo, determinar com mais precisão necessidades de ampliação da infra-estrutura.



(a) (b)

Figura 4. Número de jobs executados em um determinado período de tempo.

Na Figura 5 é possível observar 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 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*).

5. 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.

O serviço GUACS foi implementado utilizando como base o *framework* Muse [Muse 2005] – versão 1.0, da Apache Software Foundation. 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.

```
<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 5. Formato de 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>2005-09-13 17:30:50</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 6. Formato de notificação recebida pela aplicação de gerenciamento.

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).

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 2005].

A aplicação de gerenciamento, quarto componente da arquitetura, também foi desenvolvida em Java fazendo uso da biblioteca JFreeChart (para gerar gráficos) [JFreeChart 2005]. Atualmente estamos trabalhando, também, no desenvolvimento de uma versão da aplicação para operar de forma integrada à plataforma de gerenciamento HP Open View.

6. Avaliação Experimental

Esta seção apresenta uma avaliação experimental realizada com a implementação da arquitetura. Essa arquitetura foi, inicialmente, instanciada em um ambiente real de grade, com o objetivo de avaliar a capacidade da arquitetura em (a) prover uma visão global e uniforme do uso da infra-estrutura montada e (b) possibilitar a definição e a verificação

de políticas de divulgação das informações coletadas. Em seguida, foi conduzido um experimento visando a identificar os tempos consumidos pelos componentes da arquitetura e a caracterizar os atrasos acumulados desde que um evento é observado por um *publisher* até o recebimento da respectiva notificação pela aplicação de gerenciamento. Esses dois momentos da avaliação são detalhados nas sub-seções a seguir.

6.1. Instanciação da Arquitetura

Instanciou-se uma infra-estrutura real de grade, como ilustra a Figura 7, constituída por três domínios administrativos: uma universidade (*University A*) e duas empresas (*Company B* e *Company C*).

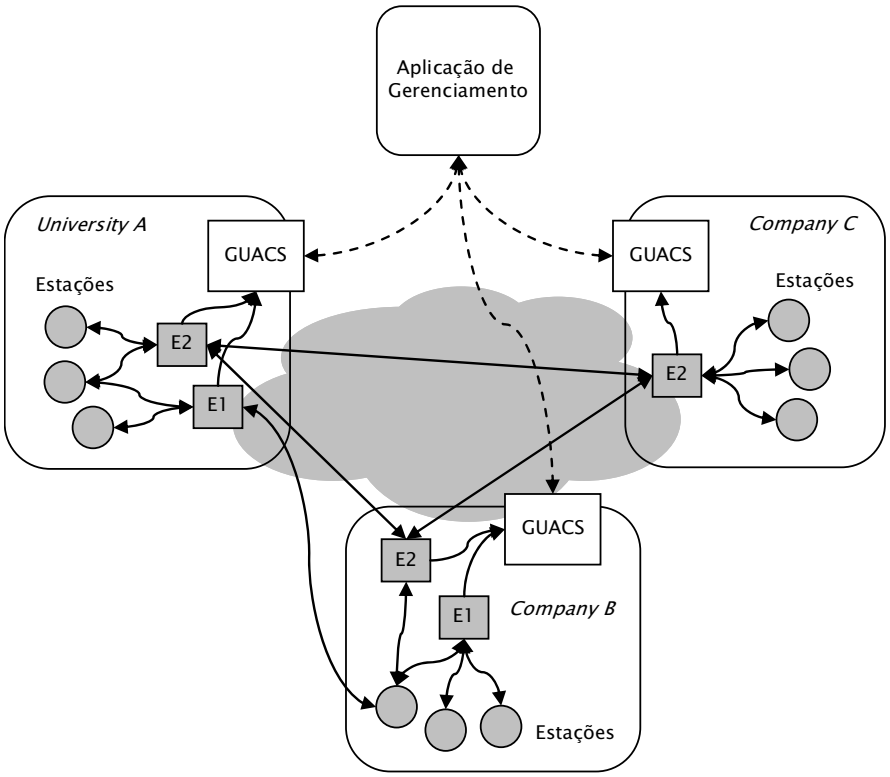


Figura 7. Infra-estrutura de grade utilizada para a avaliação da arquitetura.

University A e *Company B* têm disponível, em seus domínios, escalonadores Globus (representados na figura por E1) e compartilham estações aptas a executarem aplicações oriundas desse sistema. Outra tecnologia presente na infra-estrutura é o OurGrid. Os escalonadores OurGrid (representados por E2) formam uma rede *peer-to-peer* para o compartilhamento de recursos entre as três instituições. Foram utilizadas três estações (recursos) 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. O serviço de autorização instanciado em *University A* foi configurado com o conjunto de políticas definido na Tabela 3 (Seção 4). Já *Company B* e *Company C* usaram esse mesmo conjunto de políticas com a diferença de que o papel *Enterprise* nessas instituições foi modificado para permitir subscrição à notificação de alteração das propriedades JOBSTARTED, JOBCURRENTSTATUS, JOBCOMPLETED, JOBABORTED e JOBFAILED.

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*, isto é, uma aplicação paralela cujas tarefas são independentes umas das outras. A aplicação foi ajustada para ser disparada tanto a partir dos escalonadores Globus quanto a partir dos escalonadores OurGrid.

Para obter uma visão global e homogênea do uso da grade, primeiramente realizou-se subscrições – às propriedades JOBSTARTED, JOBCURRENTSTATUS, JOBCOMPLETED, JOBABORTED e JOBFAILED – junto aos serviços de contabilização e caracterização dispersos pelo ambiente. Essas subscrições foram feitas a partir de uma aplicação de gerenciamento posicionada em *University A*. A aplicação a ser executada na grade foi, então, disparada, demorando 32 dias e 6 horas para ser finalizada. Durante o decorrer da computação dos 16 mil *jobs* que compunham a aplicação, foi possível acompanhar o uso da infra-estrutura na medida em que notificações reportando o início e o termino da execução de *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 as tarefas estavam sendo submetidas, o estado final de suas computações e os recursos que haviam consumido – identificou-se uma situação inesperada: uma das estações disponíveis no domínio *University A* 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. Vários gráficos foram gerados pela aplicação de gerenciamento. Algumas das informações gerenciais obtidas foram: (a) número de tarefas computadas em um determinado período de tempo; (b) comparação do tempo de execução de determinadas tarefas e o estado final dessas tarefas; (c) visualização das estações onde determinadas tarefas foram computadas e os recursos que consumiram dessas estações; (d) comparação do tempo em que as estações passaram processando tarefas; e (e) número de estações disponíveis para computarem tarefas durante um determinado período de tempo.

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 *University A*, foi possível receber em tempo "quase real" notificações geradas no próprio domínio *University A*, bem como nos domínios *Company B* e *Company C*. Ao realizar as mesmas subscrições a partir do domínio *Company B*, as notificações foram geradas com um número menor de valores associados, uma vez que, naquele momento, as informações estavam sendo fornecidas a um domínio associado a um papel mais restritivo (*Enterprise* ao invés de *University*). O mesmo ocorreu com a recuperação de informações históricas (propriedade JOBHISTORY). As solicitações oriundas do domínio *University A* foram respondidas com registros UR completos, ao passo que as solicitações partindo do domínio *Company B* foram respondidas com registros incompletos (com apenas 9 de um total de 19 valores disponíveis).

6.2. Avaliação de Desempenho

Para realizar uma avaliação preliminar de desempenho da arquitetura foi usado o cenário descrito na seção anterior, restringindo o experimento a um *publisher*, a uma instância de GUACS e do serviço de autorização e à aplicação de gerenciamento. Cada um desses componentes executou em uma estação com CPU Pentium4 de 2.4 GHz, bi-processada, 1 GB RAM, interligadas por um *switch* 10/100 Mbps e sem comunicação com outras redes. O *publisher* foi submetido ao pior caso possível, isto é, eventos foram propositalmente

inseridos sequencialmente, sem intervalo de tempo entre eles no arquivo de *log* monitorado.

O objetivo do experimento foi analisar a *reatividade* do protótipo para geração de informações de gerenciamento. Para tal, buscou-se medir o tempo de processamento dos componentes da arquitetura, bem como caracterizar o atraso fim-a-fim associado à geração e à notificação de eventos. As medições realizadas, após a instrumentação dos componentes, foram as seguintes:

- tempo total decorrido desde o momento em que a ocorrência de um evento é detectada pelo *publisher* até o instante após a mensagem de atualização ter sido disparada (T_1);
- tempo total decorrido desde o momento em que a mensagem de atualização é disparada até o momento após ela ser recebida pelo serviço de contabilização e caracterização (T_2);
- tempo total decorrido desde o momento em que a mensagem de atualização é recebida pelo serviço de contabilização e caracterização até o momento após as informações terem sido inseridas na base de dados (T_3);
- tempo total decorrido desde o momento em que o serviço de contabilização e caracterização dispara uma mensagem de notificação até o momento após ela ter sido recebida pela aplicação de gerenciamento (T_4);
- tempo total decorrido desde o momento em que o serviço de autorização recebe uma requisição até o momento após a análise das políticas de divulgação ser efetuada (T_5).

Para obter resultados estatisticamente significativos, o experimento foi repetido 400 vezes. Os valores médios obtidos para T_1, T_2, T_3, T_4 e T_5 foram 16,96, 125,26, 5,06, 98,34 e 44,32 ms, respectivamente. O desvio padrão observado não foi superior a 3 ms em nenhum desses tempos.

Analisando os tempos coletados, percebe-se que a sobrecarga introduzida pelos componentes da arquitetura é aceitável. Os maiores tempos observados referem-se a T_2 e T_4, os quais envolvem a rede de comunicação (extrapolando o domínio do protótipo). O tempo T_5, terceiro maior observado, reflete o tempo de análise das políticas de divulgação de informações. Esse tempo é proporcional ao número de políticas empregadas.

Em relação ao atraso fim-a-fim, isto é, ao intervalo decorrido entre o momento em que um evento é detectado pelo *publisher* até o momento em que a notificação é recebida pela aplicação de gerenciamento ($T_1 + T_2 + T_3 + T_4$), o tempo mensurado foi de 245,62 ms. Esse tempo pode ser considerado bastante satisfatório, uma vez que possibilita à aplicação de gerenciamento (a) ser informada quase que imediatamente sobre eventos importantes observados na infra-estrutura de grade, bem como (b) reagir rapidamente, por exemplo, a descontinuações na execução das aplicações.

Ressalta-se que a arquitetura não gera carga extra na execução de aplicações no ambiente de grade, já que os *publishers* desenvolvidos estão baseados na coleta de informações que vão sendo armazenadas em arquivos de *log*, ou seja, a arquitetura não está acoplada ao *middleware* das tecnologias de grade.

7. Conclusões e Trabalhos Futuros

As soluções existentes para contabilização de uso de grades computacionais são limitadas (a) por não oferecerem uma visão global e uniforme do uso de infra-estruturas que são compostas por diferentes tecnologias de grade, (b) por não permitirem a especificação de políticas de divulgação das informações coletadas e (c) por não gerarem estatísticas sobre as aplicações que são executadas. Este artigo propôs uma abordagem, acompanhada de uma arquitetura, para a contabilização e a caracterização de uso de grades computacionais que supre essas limitações. 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 apontam que a arquitetura proposta pode ser aplicada em situações reais, fato comprovado pela implementação de um protótipo capaz de monitorar durante 32 dias a execução de uma aplicação real, do seu lançamento ao seu término. Esse protótipo também permitiu realizar um experimento para caracterizar o atraso acumulado desde que um evento é observado por um *publisher* até o recebimento da respectiva notificação pela aplicação de gerenciamento. O atraso mensurado foi considerado baixo levando em conta a aplicação real objeto de estudo deste artigo.

Como trabalhos futuros 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. Além disso, pretende-se concluir a implementação de uma GUI (*Graphical User Interface*), que permitirá aos administradores da grade interagir mais facilmente com a arquitetura e tirar maior proveito das suas funcionalidades.

Referências

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

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

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

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

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.

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

Globus Toolkit Project Home Page (2005) http://www.globus.org.

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

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

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

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

Ludwig, G., Gaspary, L. P., Cavalheiro, G. G. H. e Cirne, W. (2006) "A WSDM-based Architecture for Global Usage Characterization of Grid Computing Infrastructures", In: IFIP/IEEE International Workshop on Distributed Systems: Operations and Management. Lecture Notes in Computer Science, v. 4269, p. 124-135.

Massie, M. L., Chun, B. N. e Culler, D. E. (2004) "The Ganglia Distributed Monitoring System: Design, Implementation, and Experience", In: Parallel Computing, vol.30, n.7, p. 817-840.

MUSE Project Home Page (2005) http://ws.apache.org/muse.

Newman, H. B. et al. (2003) "MonALISA: a Distributed Monitoring Service Architecture", In: Computing in High Energy and Nuclear Physics.

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

RUS Project Home Page (2005) http://www.doc.ic.ac.uk/~sjn5/GGF/rus-wg.html.

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) Management of Web Services (WSDM-MOWS). Version 1.0. OASIS Standard. http://docs.oasis-open.org/wsdm/2004/12/wsdm-mows-1.0.pdf.

Srivastava, R.,Yin, L. e Yin, J. (2002) "Stochastic vs. Deterministic Modeling of Intracellular Viral Kinetics" Theory Biology, vol. 218, p. 309-321.

Sun's XACML (2004) Implementation Programmer's Guide. http://sunxacml.sourceforge.net/guide.html.

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

UR Project Home Page (2005) https://forge.gridforum.org/projects/ur-wg.

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.