CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

MONITORAMENTO DE RECURSOS EM AMBIENTES DE GRADE

DANIEL DA TRINDADE LEMOS

Canoas, novembro de 2006

DANIEL DA TRINDADE LEMOS

MONITORAMENTO DE RECURSOS EM AMBIENTES DE GRADE

Trabalho de conclusão apresentado à banca examinadora do curso de Ciência da Computação do Centro Universitário La Salle - Unilasalle, como exigência parcial para obtenção do grau de Bacharel em Ciência da Computação, sob orientação da Profa. DSC. Patrícia Kayser Vargas Mangan.

Canoas, novembro de 2006

TERMO DE APROVAÇÃO

DANIEL DA TRINDADE LEMOS

MONITORAMENTO DE RECURSOS EM AMBIENTES DE GRADE

Trabalho de conclusão aprovado como requisito parcial para obtenção do grau de Bacharel em Ciência da Computação do Centro Universitário La Salle - Unilasalle, pela seguinte banca examinadora:

Prof. Me. Marcos Ennes Barreto
Centro Universitário La Salle - Unilasalle

Prof. Me. Mozart Lemos de Siqueira
Centro Universitário La Salle - Unilasalle

Prof. Dsc. Patrícia Kayser Vargas Mangan
Centro Universitário La Salle - Unilasalle

Canoas, novembro de 2006

RESUMO

Em um ambiente de computação em grade, a coleta e o processamento de informações de monitoramento de recursos apresenta desafios relacionados a grande heterogeneidade e questões de escalabilidade. Essas informações podem ser usadas tanto para auxiliar ferramentas de descoberta quanto de escalonamento de tarefas. Além disso, podem auxiliar ferramentas de tomada de decisão gerencial. Após a análise de ferramentas disponíveis, nota-se que muitas disponibilizam dados sobre utilização de CPU e memória, mas quase nenhuma disponibiliza informações de software. Assim, este trabalho busca um modelo de monitoramento de recursos de hardware e software, que contemple as questões de escalabilidade e heterogeneidade,para ambientes de grade. Um protótipo foi implementado para o sistema operacional Windows e testado em laboratórios do Banrisul. As avaliações mostraram baixa intrusão no sistema. As informações de monitoração são disponibilizadas com documentos XML para facilitar a integração com outras ferramentas.

Palavras-chave: Computação em grade, monitoramento, *Web services*.

Resource Monitoring in Grid Environments

ABSTRACT

In a grid computing environment, to collect and to process information concerning resource monitoring is challenging due to heterogeneity and scalability issues. This information can be used to help both discovery and task scheduling tools. Besides, it can help decision making tools. After analysing the available tools, we can observe that several provides information about CPU and memory usage, but almost none provide software information. Thus, this work aims at proposing a hardware and software resource monitoring model for grid environment, which deals with scalability and heterogeneity issues. A prototype was implemented for Windows operating system, and tested on Banrisul labs. The evaluation showed low intrusion. The monitoring information is provided as XML documents to facilitate integration with other tools.

Keywords: Grid Computing, Monitoring, Web Services.

SUMÁRIO

LISTA DE SÍMBOLOS E ABREVIATURAS

LISTA DE FIGURAS

LISTA DE TABELAS

1 INTRODUÇÃO

Hoje em dia, apesar dos avanços atingidos pela computação, algumas aplicações ainda exigem a necessidade maior do que o processamento suportado por computadores individuais tornando o processamento paralelo e distribuído uma necessidade. Além disso, o alto custo de máquinas poderosas reforça a importância do escalonamento de processos entre máquinas através de redes locais ou clusters. Em especial nas instituições (acadêmicas e/ou comerciais) de pequeno e médio porte, seria possível ganhar um maior poder de processamento de informações caso houvesse um compartilhamento de seus recursos através deste tipo de processamento.

A possibilidade de que máquinas de instituições distintas disponibilizem seus recursos computacionais para solucionar os problemas de outros usuários e instituições através de em uma rede de computadores integrada fez surgir a idéia da Computação em Grade (*Grid Computing*) (de Camargo et al., 2004), (Foster et al., 2001), (Foster and Kesselman, 2003).

Um sistema de Computação em Grade pode ser definido como uma infra-estrutura capaz de interligar e gerenciar diversos recursos computacionais distribuídos por uma rede de computadores de maneira a oferecer, ao usuário, uso intensivo de recursos e aplicações distribuídas (Chin et al., 2004). Em grades existem potencialmente uma quantidade grande de usuários, recursos e aplicações sendo administradas por diferentes domínios administrativos.

Neste contexto computacional, uma das linhas de pesquisa é o monitoramento. O monitoramento dos recursos existente na grade serve principalmente como auxílio a ferramentas de escalonamento de processos, a fim de definir os recursos que serão utilizados por cada tarefa.

Este texto apresenta a proposta de um novo módulo ou serviço a ser usado por um sistema gerenciador de aplicações para ambiente em grade. Esse serviço pode ser usado por sistemas desenvolvidos a partir de plataforma como EasyGrid (de P. Nascimento et al., 2005), GRAND (*Grid Robust Application Deployment*)

(Mangan, 2006) ou Globus (Globus, 2006). O Globus fornece um serviço de monitoração e descoberta chamado MDS (*Monitoring and Discovery Service*), mas alguns requisitos levantados neste trabalho parecem não ser totalmente atendidos por esse serviço. O objetivo é obter e publicar informações de hardware, algo que é essencial e que já existem ferramentas que disponibilizam, bem como disponibilizar informações de software, o que permitirá um uso geral, podendo ser utilizado também por administradores de rede e gerentes.

Deste modo, um dos objetivos deste trabalho é estudar técnicas e modelos de monitoramento de recursos, a fim de determinar uma forma de monitoramento de recursos, que contemple monitoração tanto de aspectos estáticos quanto dinâmicos de hardware e software para um ambiente de grade, para auxiliar escalonadores de recursos e também administradores de ambiente em grade.

Este trabalho é classificado como um estudo de caso de natureza aplicada, pois testa modelos de monitoramento para um ambiente específico. Ele está sendo desenvolvido dentro do contexto do modelo GRAND (Mangan, 2006). O GRAND (*Grid Robust Application Deployment*) é um modelo de gerenciamento hierárquico de aplicações em ambiente de grade. Seu objetivo é tratar de forma escalável a submissão e o monitoramento de aplicações que disparam um número muito grande de tarefas (centenas ou milhares).

A integração de informação de monitoramento, obtidas com este trabalho, no escalonamento do GRAND, permitirá a realização de um escalonamento mais eficaz.

O restante deste texto apresenta-se organizado do seguinte modo. Inicialmente, apresentam-se o Capítulo 2 sobre Monitoramento em Grade, onde será abordado as principais ferramentas de monitoramento existentes hoje em grades computacionais. Depois terá o Capítulo 3 com o modelo, onde tratará principalmente das premissas, dos dados monitorados e da modelagem e protótipo. Depois, o Capítulo 4 analisa os resultados obtidos com os experimentos efetuados. Finalmente terá o capítulo 5 concluido este texto também abordando os trabalhos futuros.

2 MONITORAMENTO EM GRADE

Nos últimos anos, houve um significativo aumento no uso de redes de computadores em ambientes comerciais e de pesquisa. Os computadores que normalmente compõem essas redes são heterogêneos e podem encontrar-se distribuídos através de prédios, cidades e até países diferentes. Esses equipamentos precisam estar conectados através de uma infra-estrutura de hardware e software que pode ser tanto um sistema distribuído ou uma grade computacional (*grid computing*).

Segundo Mattos (de Mattos, 2003) uma grade computacional é uma infra-estrutura de hardware e software que provê acesso seguro, consistente, de forma distribuída e a custo baixo.

Segundo (Dantas, 2005) grade é um ambiente computacional de alto desempenho, o qual é caracterizado por prover o compartilhamento geograficamente distribuído de serviços por organizações distribuídas geograficamente.

Foster et al. (Foster et al., 2001) define uma grade como um sistema que:

- coordena recursos de diferentes fins (aplicações científicas, comerciais, desktop, etc.);
- usa interfaces e protocolos padronizados, abertos e para propósitos gerais;
- oferece QoS (qualidade de serviço) não triviais: segurança, autenticação, escalonamento de tarefas (distribuição do processamento), disponibilidade, tempo de resposta.

Além disso, existem três principais aspectos que caracterizam grades computacionais (Mangan, 2006):

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

- Escalabilidade (*scalability*): uma grade pode crescer de poucos recursos para milhões. Isto levanta o problema da potencial degradação do desempenho à medida que o tamanho de uma grade cresce. Conseqüentemente, aplicações que requerem um grande número de recursos dispersos geograficamente devem ser projetados para serem extremamente tolerantes a latência;

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

Desta forma um ambiente de grade necessita prover ferramentas que permitam monitorar seus componentes.Segundo Bona (Bona, 2004) o monitoramento tem que prever a dinamicidade do ambiente que a grade impõe. Ainda podem ser implementados mecanismos de tolerância a falhas que precisam conhecer o estado do sistema.

Neste contexto, uma das questões a serem analisadas em uma grade computacional é o monitoramento, pois serve para permitir uma melhor gerência do ambiente. Segundo HOLLINGSWORTH e TIERNEY (Hollingsworth and Tierney, 2004) o monitoramento da grade é a medição e a publicação do estado de um componente em um ponto particular da grade em um determinado instante de tempo. Existem várias ferramentas para monitoramento, como por exemplo Netlogger (NetLogger, 2006), R-GMA (R-GMA, 2006), Gridbus (GridBus, 2006), Remos (Remos, 2006), EasyGrid (de P. Nascimento et al., 2005), bem como para visualização de dados de monitoramento como Paje (Neves et al., 2004) e Monalisa (MonaLisa, 2006). No entanto, nas próximas seções, iremos analisar apenas Ganglia, NWS-Apples, Hawkeye-Condor e WebMDS-Globus pois são as ferramentas mais citadas na literatura.

O restante deste capítulo trará mais detalhes sobre Monitoramento em Grade a Seção 2.1, apresenta alguns conceitos adicionais sobre monitoramento. Os principais ambientes de monitoramento são apresentados na Seção 2.2 e analisados na Seção 2.3.

2.1 Monitoramento

Segundo Jain (Jain, 1991) um monitor é uma ferramenta utilizada para observar as atividades de um sistema. Em geral, os monitores observam o desempenho do sistema, coletam estatísticas de desempenho, analisam dados e mostram resultados.

O monitoramento não é utilizado apenas por programadores interessados em analisar o desempenho do sistema, mas por programadores em geral e por administradores de rede.

Duas das razões citadas por jain (Jain, 1991) para monitorar um sistema são as seguintes:

- um programador de sistema pode usar os dados monitorados para encontrar um segmento de software freqüentemente usado e otimizar seu desempenho;
- um administrador pode usar as medidas de utilização de recursos monitorados para detectar gargalos (*bottlenecks*) de desempenho.

No monitoramento de software, encontra-se um grande problema nas corporações: como inibir a utilização de softwares piratas ou sem licenças, através da monitoração dos softwares instalados. Também seria importante determinar a localização e tempo de utilização de softwares estratégicos para o funcionamento da corporação.

Além disso, em uma grade acadêmica onde tipicamente há uma maior heteregeidade, é possível que um determinado software somente esteja disponível em um determinado nó da grade. Essa informação deve ser então utilizada para o escalonamento de tarefas que possuam restrições de execução em uma determinada plataforma e/ou necessitem de um software específico para execução.

A ferramenta de monitoramento pode ser usada também para auxiliar uma ferramenta de descoberta onde será possível fazer busca por recursos identificados por exemplo como máquinas com uma determinada quantidade de memória ou que possua um determinado software.

As informações de monitoramento de hardware e software devem ser coletadas máquina a máquina, mas normalmente são centralizadas nas redes locais ou *clusters*. Além disso, periodicamente esses dados são enviados a um elemento central para permitir uma visão global dos recursos da grade. Com relação a este envio dos dados coletados, deve-se levar em conta a largura de banda para não sobrecarregar a rede e não afetar o desempenho da mesma. O envio periódico das informações deve ser feito para monitorar mudanças de características de hardware e software, enviando as diferenças e guardando um histórico para análise das mudanças.

Deste modo, a motivação para o estudo de monitoramento de recursos em ambiente de computação em grade é, considerando a distribuição e alocação dos recursos computacionais, disponibilizar dados para:

- Auxílio na tomada de decisão dos administradores na compra de determinado software ou hardware, bem como pela renovação ou não de licenças de software;
- Permitir uma boa alocação de recursos por parte de sistemas gerenciadores de recursos (RMS ou *resource management system*) bem como um bom escalonamento de tarefas por gerenciadores de aplicação em contexto de computação em grade.

2.2 Trabalhos Relacionados

Nas subseções que seguem será abordado os principais trabalhos na área da computação em grade que tratam de monitoramento de recursos. Os trabalhos foram escolhidos pois são atualmente os mais referenciados na literatura.

2.2.1 Ganglia

É um sistema de monitoramento distribuído e escalável para sistemas de computação de alto desempenho, como clusters e grids. Este sistema é baseado em uma arquitetura hierárquica focada em federações de *clusters*. A implementação de Ganglia é robusta e já foi portada para vários sistemas operacionais e arquiteturas, de modo que esta ferramenta é atualmente utilizada em um grande número de *clusters* em todo o mundo(Ganglia, 2006).

Para armazenamento e visualização dos dados monitorados, Ganglia utiliza o sistema RRDtool(*Round Robin Database Tool*). Este sistema permite o armazenamento de seqüências temporais de dados de forma compacta em um banco de dados circular, de tamanho constante. A partir dos dados armazenados, RRDtool gera gráficos que mostram a evolução de uma ou mais métricas ao longo do tempo.

A Figura 2.1 mostra a arquitetura do Ganglia indo desde os nós onde estão os *Deamons*, responsáveis pela coleta de informações, até o envio dos dados por Multicast UDP, para a base de Dados, para uma posterior consulta através da Interface do Ganglia.

A Figura 2.2 ilustra a interface Web do Ganglia para visualização das métricas monitoradas. Nesta interface, observa-se na parte superior um menu de opções onde se pode selecionar a métrica e a granularidade de tempo a ser visualizada. Já na parte central estão as métricas globais do cluster e, na parte inferior, a visualização dos estados de cada nó com relação a uma métrica escolhida no menu superior.

2.2.2 Condor e Hawkeye

Condor é um sistema de gerenciamento de recursos surgido em 1984, desenvolvido pela Universidade Wisconsin em Madison nos EUA. Ele é um *middleware* que detecta ciclos ociosos e com base nestas informações aloca tarefas. O conjunto de recursos é chamado de *Condor Pool*. Um *pool* pode ser utilizado por diversas instituições, podendo dar origem a uma grade computacional.

O funcionamento do *Condor pool* está ilustrado de acordo com a Figura 2.3, as aplicações são submetidas através de agentes que são responsáveis por armazenar tarefas da aplicação até encontrar um recurso adequado disponível para execução. Tanto os agentes quanto os recursos reportam seus estados para *matchmaker*, que é responsável por administrar quais os recursos são mais adequados para execução.

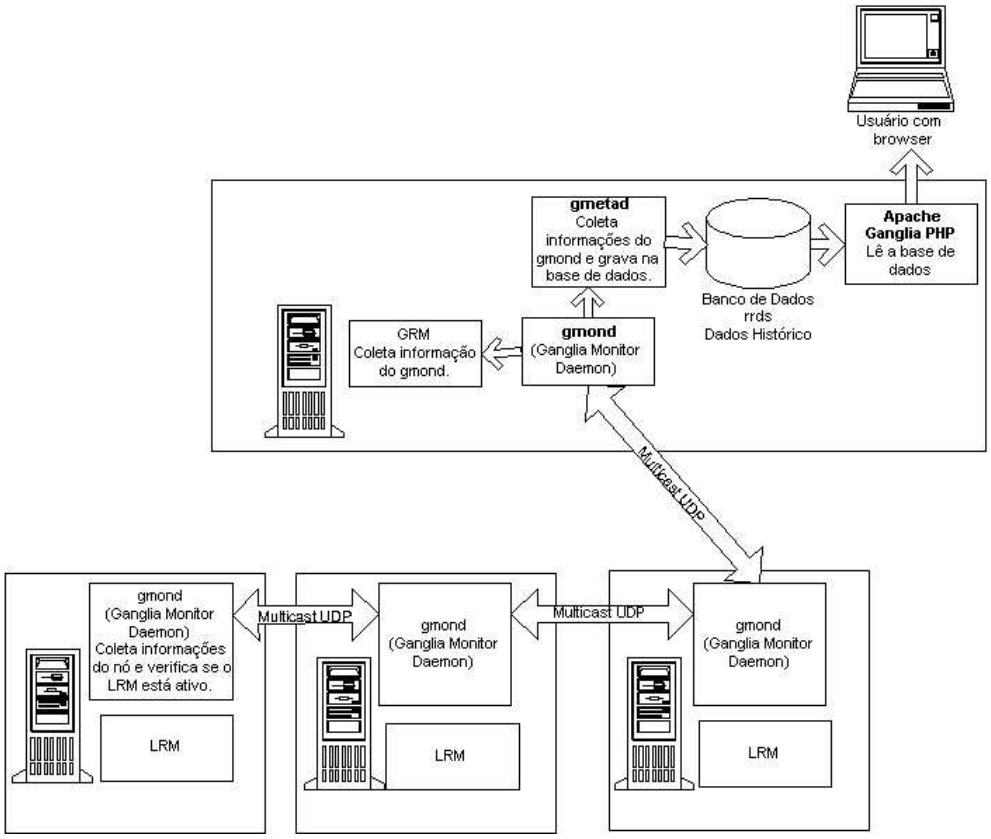


Figura 2.1: Ganglia: Arquitetura
(Massie et al., 2004)

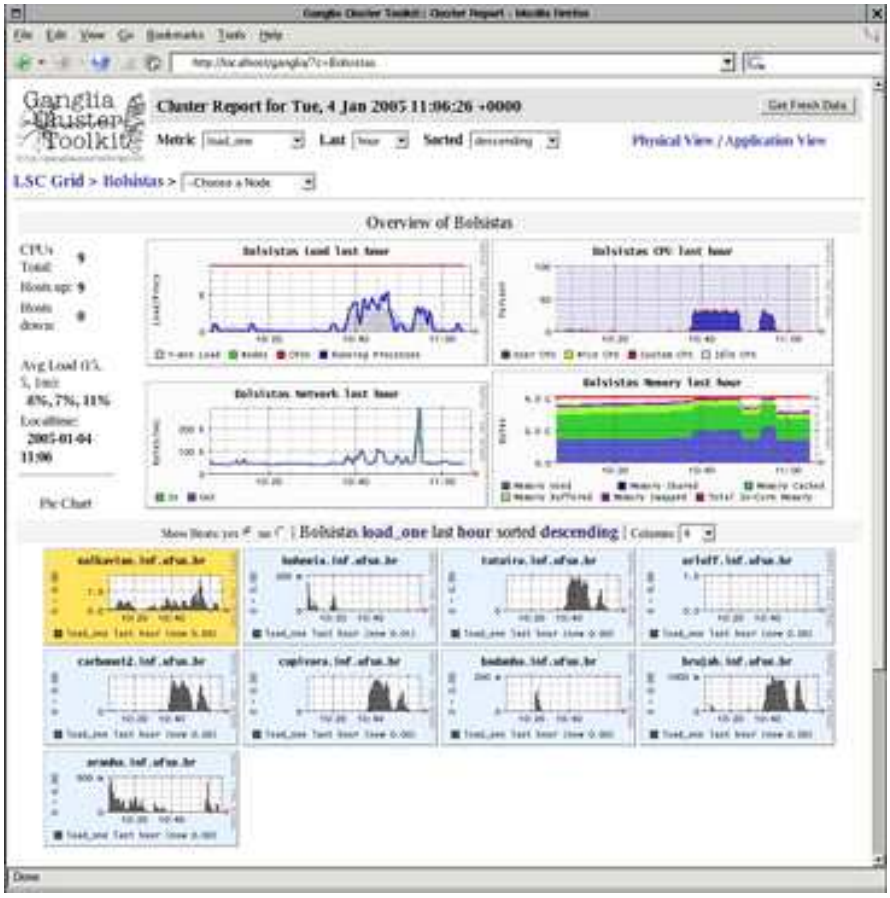


Figura 2.2: Ganglia: Interface
(Massie et al., 2004)

Para que este casamento (*matching*) de recursos ocorra, as devisões são baseadas em informações de monitoramento armazenamento em uma gerenciador central no *pool*.

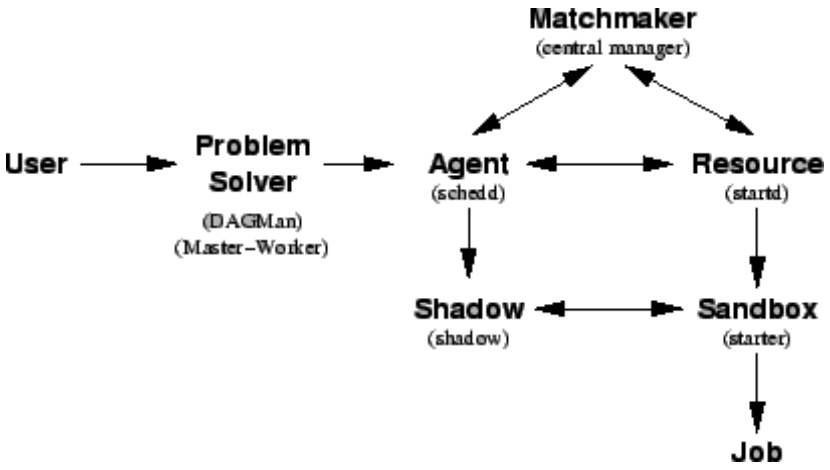


Figura 2.3: Condor: Arquitetura
(Andrade, 2006)

A Figura 2.4 mostra como é o funcionamento do mecanismo *Flocking*, que é responsável por permitir o compartilhamento de recursos e migração de tarefas entre as *Condor pool*. Ele surgiu para aumentar a capacidade dos aglomerados, com o nome de *gateway Flocking*, que faz a interligação entre *Condor pools*. Cada *gateway* é configurado com informações dos outros *pools* para que seja possível haver a associação entre os mesmos.

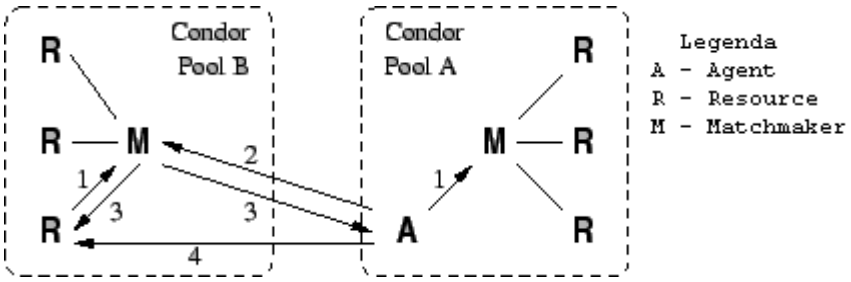


Figura 2.4: Condor *Flocking*
(Andrade, 2006)

Este compartilhamento é feito através do *gateway*, preservando a estrutura dos dois *Pools*. Cada um dos nós trocam informações sobre os participantes de seus *pools* quando é detectado um recursos ocioso, imediatamente esta informação é passada para o *pool* que repassa para o *matchmaker* remoto. Uma característica importante é que esse compartilhamento pode ser unidirecional.

Hawkeye (Hawkeye, 2006a) utiliza as tecnologias existentes no projeto Condor incusive as classAds, prove mecanismos de coleções e histórico e utiliza as informações dos computadores. O sistema Hawkeye monitora vários atributos e coleções do sistema. O mecanismo de monitoração é muito usado por administradores para o gerenciamento do sistema.

A figura 2.5 mostra a interface do sistema Hawkeye, onde é possível monitorar os recursos e verificar a execução de tarefas em cada nó.

Na parte superior da janela mostra a lista de recursos como problemas, aplicações em uso e outras informações. note que essa interface fornece apenas informações em modo texto.

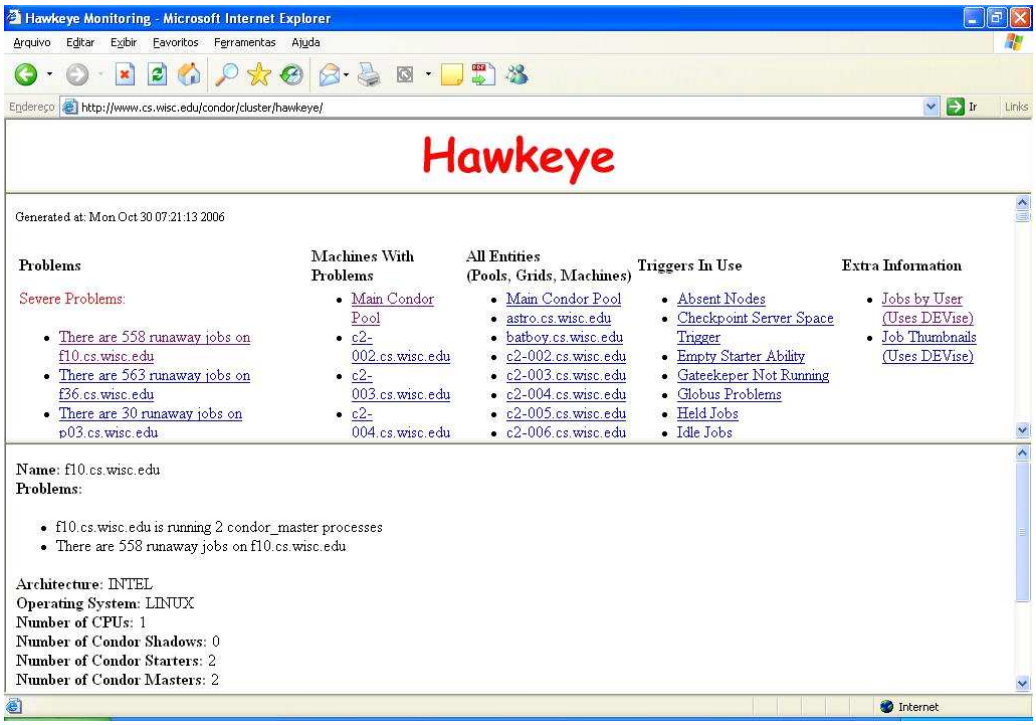


Figura 2.5: Hawkeye: Interface
(Hawkeye, 2006b)

2.2.3 Globus e WebMDS

O projeto Globus atualmente é uma iniciativa que envolve instituições de pesquisa com o apoio de grandes empresas como IBM e a Microsoft. As principais instituições de pesquisa são o ANL (Laboratório Nacional Argone), Universidade de Chicago, USC (Universidade do Sul da Califórnia) e o laboratório de computação de alto desempenho da Universidade do Norte de Illinois, todas estas instituições estão situadas nos Estados unidos, além da Universidade de Edimburgo. O sistema projeto Globus no contexto da computação em Grade é denominado Globus *Toolkit* (Globus, 2006) e prove uma série de funcionalidades na forma de serviços.

O sistema Globus é construído sobre uma arquitetura em camadas cujos níveis mais altos são estruturados sobre os níveis mais baixos (serviços de base). O Globus possui ênfase em uma integração hierárquica dos componentes da grade e seus serviços (dos Santos, 2006).

O Globus Toolkit sofreu profundas alterações entre as versões 2 e 3 (GT3, 2006), havendo mudanças nos módulos, interfaces e protocolos.

A versão 3 forneceu a primeira implementação do OGSI 1.0, para mostrar o funcionamento do OGSI , OGSA e Globus a figura 2.6 mostra as entidades que se relacionam formando o cenário de padrões e implementação da tecnologia de *Grid de Serviços*.

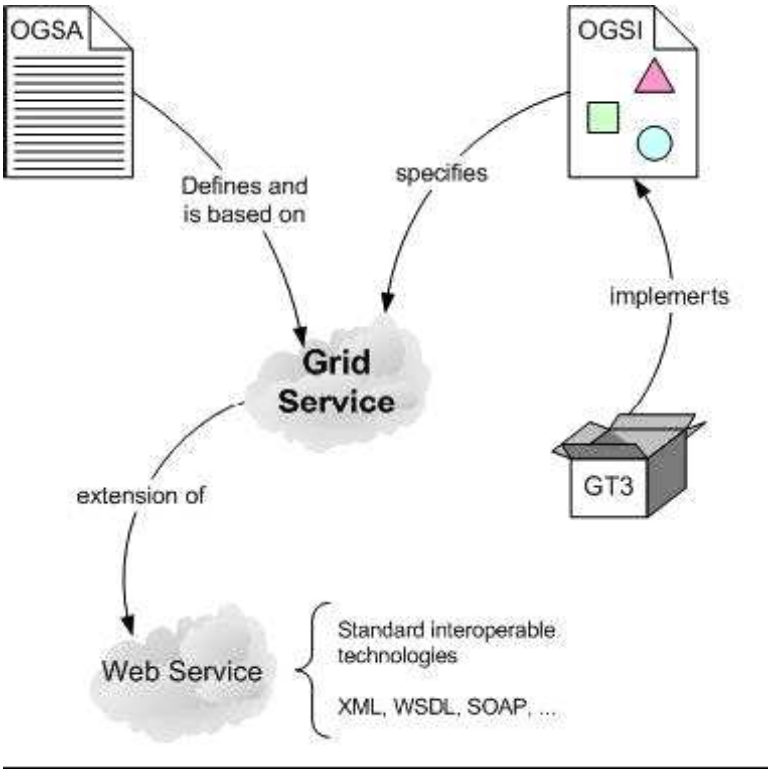


Figura 2.6: Funcionamento *Grid Service*
(GT3, 2006)

Em 2005 foi lançada a versão 4 (GT4, 2006) onde o ênfase foi pela robustez, facilidade de uso e documentação. Mas o ponto mais importante desta versão é que seja compatível com todas interfaces *Web Services*.

O MDS (*Monitoring and Discovery Service*) é um serviço oferecido pelo Globus *Toolkit* que é capaz de prover informações e status da grade e dos recursos disponíveis por exemplo rede, nós e sistemas de armazenamento.

Este serviço está disponível na versão 2 cuja implementação é baseada em LDAP utilizando os serviços de informação do Globus *Toolkit* versão 2.x. E também existe

na versão 3, que é implementado baseado em OGSI usando serviços de informação do globus *toolkit* versão 3.x.

Na versão 4 ele sofreu ainda mais alterações na implementação e disponibilização de informações. Nas 3 versões, o objetivo principal é fornecer informações para outros serviços do Globus. O WebMDS (WebMDS, 2006) é mais uma funcionalidade proporcionada pela verão 4 do Globus *Toolkit*. O WebMDS usa a interface do navegador utilizando a formatação XML efetuando a busca por resultados das pesquisas tendo como parâmetro as propriedades dos recursos no *Grid Services*. WebMDS é um *Servlet* que utiliza *plugins* para adquirir dados de documentos XML e transformando em XSL.

A Figura 2.7 mostra o funcionamento do Globus *toolkit* versão 4, utilizando o serviço do WebMDS. Cada quadrado caracteriza um nó, dentro de cada nó está rodando alguns serviços como o MDS ou GRAM. Quando o MDS está no ar ele se registra no MDS index. O MDS é o centralizador das informações e é responsável por fazer com que os módulos se comuniquem. A partir dele partem as informações que serão visualizadas através do WebMDS.

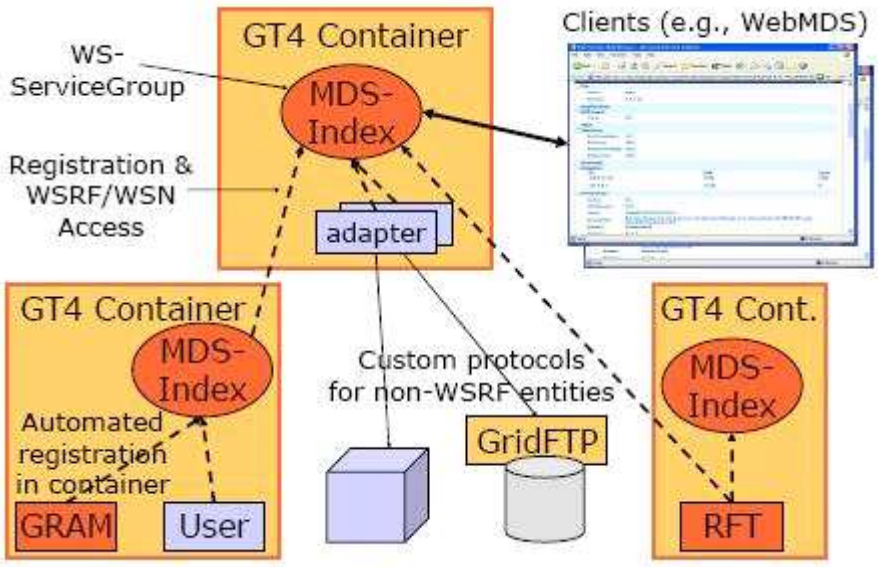


Figura 2.7: Globus *Toolkit* 4
(GT3, 2006)

2.2.4 GrADS e NWS

Segundo (Wolski et al., 1999), NWS (*Network Weather Service*) é um sistema que monitora e dinamicamente provê estimativas de desempenho de recursos computacionais. O processo de monitoramento é feito através de sensores que medem periodicamente o estado dos recursos. As informações coletadas podem ser requisi-

tadas diretamente pelos escalonadores, que utilizam as informações para melhorar o escalonamento.

Além da informação sobre o desempenho de um determinado recurso, uma heurística de escalonamento pode ser construída com previsões de desempenho com base no histórico obtido com a monitoração. Assim, através de métodos numéricos as informações armazenadas sobre o desempenho são processadas para gerar estimativas de desempenho para determinar o que pode ser utilizado por um determinado nó em um período de tempo.

GrADS (*Grid Application Development Software*) é um sistema de escalonamento que trata de um ambiente heterogêneo de computação, e em especial, para invocar outras ferramentas de computação em grade. Para obtenção das máquinas existentes é usado o *resource selector* que acessa o globus MDS (*Monitoring and Discovery Service*) para obter a lista das máquinas ativas, com a lista das máquinas ativas e utilizado serviços do NWS para busca das informações de sistema das máquinas. O *applications manager* invoca o componente *performance modeler* utiliza as informações das máquinas para organizar a lista de execução.

2.3 Análise

As ferramentas foram analisadas e os dados sumarizados em duas tabelas.

Nas Tabelas 2.1 e 2.2 três símbolos foram empregados. Os "X" indicam presença da característica, enquanto um "-" indica a ausência da característica. Já o símbolo "+/-" indica que a ferramenta atende parcialmente o item.

Pode-se observar que todas as ferramentas suportam Linux, e somente três também suportam Windows. Quase todas foram implementadas em "C", mas somente duas foram implementadas em Java. Vale destacar que a ferramenta WebMDS é ferramenta mais recente.

A Tabela 2.1 resume algumas das principais características de implementação das ferramentas de monitoramento analisadas. Na segunda coluna podemos observar as plataformas que as ferramentas executam, na terceira coluna temos a linguagem nas quais foram implementadas, na quarta e quinta temos, respectivamente, a coluna que diz se a ferramenta integra a um outro *framework* e ferramentas de monitoramento.

A Tabela 2.2 apresenta características adicionais relacionadas aos dados coletados. A primeira coluna indica a ferramenta analisada. Na segunda coluna,indica se a ferramenta monitora CPU, Memória e/ou softwares. A última coluna mostrará se a ferramenta devolve os dados monitorados de uma forma gráfica ou textual.

Na Tabela 2.2 todas as ferramentas fornecem dados como porcentagem utilização de CPU apenas uma fornece dados sobre a porcentagem de utilização de memória.

Quanto a softwares somente uma obtém algum tipo de informação sem fornecer

Tabela 2.1: Ferramentas analisadas: Características da implementação

Ferramenta	**Plataforma**	**Linguagem**	**Integração com middleware de Grade**	**Integração com Ferramenta de Monitoramento de Grade**
Hawkeye	Linux/Windows	C/Perl	Condor	-
WebMDS	Linux/Windows	Java	Globus	-
Ganglia	Linux	C	-	LibRastro
NWS	Linux	C	GrADS	MDS
MDS	Linux/Windows	Java	Globus/GrADS	-

Tabela 2.2: Ferramentas analisadas: Dados Coletados

Ferramenta	**Informações**			**Visual. Dados**	
	CPU	Mem.	Software	Gráfica	Textual
Hawkeye	X	X	+/-	X	X
WebMDS	X	-	-	X	-
Ganglia	X	-	-	X	-
NWS	X	-	-	+/-	-

informação de quais estão instalados a localização ou versão, o que também seria fundamental para um melhor escalonamento de tarefas.

Todas as ferramentas analisadas disponibilizam informações pertinentes ao escalonamento, mas não atendem na totalidade a expectativa dos escalonadores, pelo fato de fornecer as informações como utilização de CPU e memória, mas não trazem informações de softwares como quais são e a localização. Também não se preocupam com o tráfego destas informações podendo sobrecarregar a rede com este tipo de informação. Estas questão são tratadas pelo modelo proposto no próximo capítulo.

Quase todas disponibilizam algum tipo de forma de visualização seja gráfica ou textual. No entanto, essas interfaces são limitadas, indicando um outro problema em aberto, mas que não será tratado neste trabalho.

3 MODELO DE MONITORAMENTO DE RECURSOS

O modelo consiste em obter informações de software e hardware de forma pouco intrusiva e que utilize pouca banda não sobrecarregando a rede na hora de fazer o envio das informações coletadas para outras máquinas. A intrusão é baixa com o percentual de CPU aumentando somente no momento da obtenção dos dados. Para evitar a sobrecarga da rede, as informações serão enviadas em tempos distintos. Cada nó enviará em um determinado período para que auxilie e não entre em conflito com processos mais importantes. Além disto, as informações são enviadas por completo somente na primeira coleta,e posteriormente são enviadas somente as alterações ou atualizações. Nas seções que seguem será tratado as premissas para execução do modelo, a visão geral, os dados que serão monitorados, a modelagem e o protótipo.

3.1 Premissas

O ambiente a ser monitorado consiste em uma grade computacional. O administrador de cada nó da grade pode estabelecer diferentes prioridades para execução de processos nos computadores. Um nó da grade é composto por recursos (computadores) onde pode estabelecer diferentes prioridades para execução de processos nos computadores bem como para uso dos canais de comunicação.

Como toda a grade ela é heterogenea por natureza, considerando que sua estrutura de nós pode ser dinâmica. Normalmente a ligação e o conjunto de nós são relativamente estáticos. Já internamente os nós são bastante dinâmicos. Muitas máquinas permanecem desligadas por determinados períodos ou fora do domínio por um período.

Assim, pode-se assumir que os recursos mudam, mas os nós são mais ou menos estáticos. Formando uma estrutura padronizada, o que facilita possíveis suportes e manutenção a hardwares com defeito.

O recurso computacional é um computador cujo ambiente (hardware e/ou software) fornece uma forma de coleta de informações como o sistema operacional instalado assim como a data de instalação do mesmo e a última data de *boot*. Também deve ser possível coletar as seguintes informações: usuário logado no momento da coleta, dados do processador como a velocidade de *clock*, tipo de processador e versão, memória por cada slot e qual o tipo e velocidade da mesma, o endereço MAC, endereço IP e o domínio que o nó está conectado. Os dados referente a software incluem onde estão instalados, versão e data de instalação e também a medição da utilização de alguns softwares configurados através do servidor.

Todos os computadores de um nó da grade são capazes de se comunicar com pelo menos um computador comum que centraliza as informações coletadas. Esse computador comum deve ser grade enviará pelo menos a um servidor central as informações coletadas onde estas serão processadas e inseridas em um repositório.

3.2 Visão geral do modelo

Este trabalho propõe um modelo de monitoramento hierárquico. A Figura 3.1 mostra um possível cenário que ilustra os diferentes níveis de componentes desta hierarquia. Em cada máquina um sensor é instalado para coletar os dados do recurso. A partir do momento que a máquina é ligada, o sensor começa a monitorar os softwares usados e a coleta dos dados referentes a hardware. Muitas destas informações são estáticas, ou seja, não alteram sem ter que desligar a máquina e portanto são coletadas uma única vez.

A figura 3.1 também mostra nos itens numerados, como vai ser o envio das informações para os servidores centrais:

1. É feita a coleta da informações de software e hardware localmente e após é feito o envio para os coletores locais;

2. Os dados coletados nos coletores locais são replicados para uma área onde o servidor de banco de dados, associados a um coletor global possa processar e guardar as informações globais;

3. Requisição de dados armazenados nos coletores, estas informações são obtidas através de consultas dinâmicas.

Cada um dos elementos e itens serão detalhados nas próximas Seções. A Seção 3.3 explica quais os dados serão coletados e em qual momento é feito o processamento das informações para envio aos servidores. Na Seção 3.5 de protótipo traz a explicação do funcionamento de cada etapa para o envio dos dados ao servidor e replicação aos servidores globais.

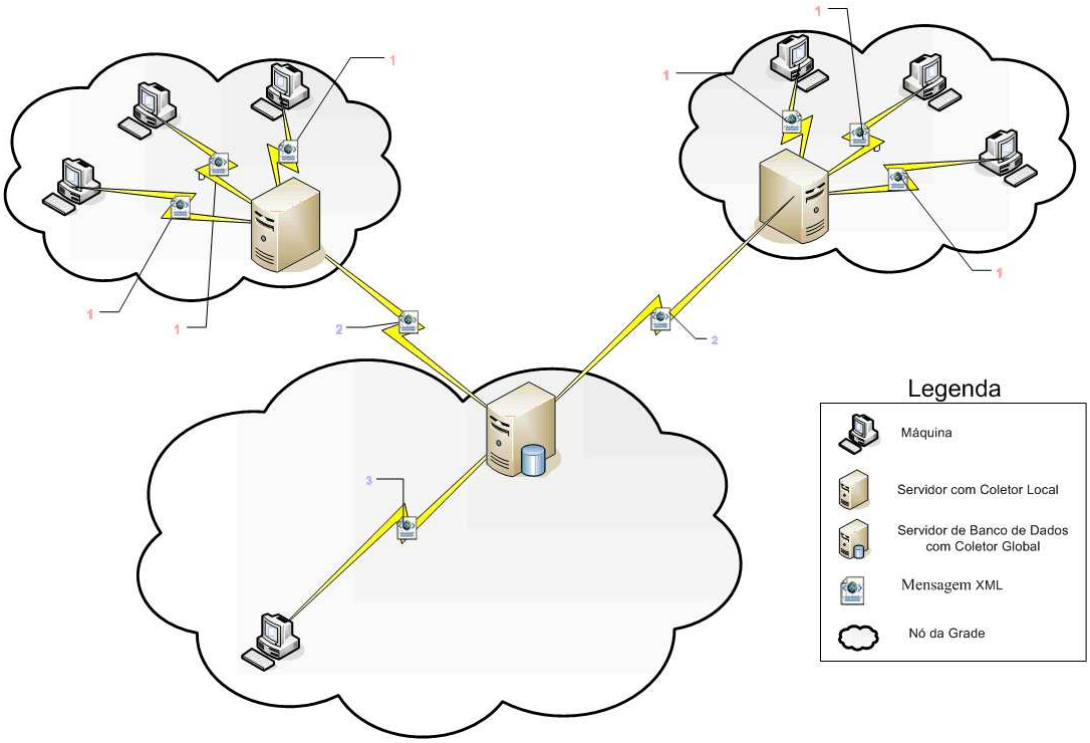


Figura 3.1: Visão geral do modelo

3.3 Dados monitorados

Os dados monitorados podem ser divididos em duas grandes categorias: os relacionados a hardware e os relacionados a software. Os dados sobre hardware tendem as ser mais estáticos enquanto os de software mais dinâmicos, mas isso não é uma regra como exposto a seguir.

A Tabela 3.1 ilustra os dados monitorados e em qual categoria estão associados. Na primeira coluna tem o tipo de dado que está sendo monitorado. Na seguintem tem o dado que esta sendo monitorado. Na terceira tem a categoria de sistemas em que se utilizará a informação considera-se duas categorias de hardware que são dados que não alteram com muita freqüência e só alteram com o desligamento da máquina e software cujo o índice de alterações são bem mais freqüênte. Na última tem o tipo de informação, que caracteriza o dado coletado como estático, são dados que não alteram enquanto a máquina estiver ligada não alteram, enquanto os dinâmicos alteram sem a necessidade de desligamento.

De um modo geral, as informações de software precisam ser monitoradas com uma freqüência maior pelo fato de alterarem com uma maior facilidade, como por exemplo.

As informações de softwares instalados, localização, data de instalação do software e versão. Outro dado a ser monitorado é o tempo de utilização de determinados

Tabela 3.1: Dados Monitorados

Tipo Dado	Dado Monitorado	Utilização	Tipo de Informação
Software	Sistema Operacional	Gerencial/Escalonamento	Estático
	Usuário	Gerencial	Dinâmico
	Data Instalação	Gerencial	Estático
	Programas Instalados	Gerencial/Escalonamento	Dinâmico
	Data Instalação	Gerencial	Estático
	Versão Programa	Gerencial/Escalonamento	Estático
Hardware	Nome Processador	Gerenciador/Escalonamento	Estático
	Velocidade Processador	Gerencial/Escalonamento	Estático
	Versão Processador	Gerencial	Estático
	Tipo Memória	Gerencial	Estático
	Localização Memória	Gerencial	Estático
	Velocidade Memória	Gerencial	Estático
	Total Memória	Gerencial/Escalonamento	Estático
	Tipo Dispositivo logico	Gerencial	Estático
	Sistema de Arquivo	Gerencial	Estático
	Tamanho	Gerencial/Escalonamento	Estático
	Espaço Livre	Gerencial/Escalonamento	Dinâmico
	IP	Gerencial/Escalonamento	Estático
	MAC Address	Gerencial/Escalonamento	Estático
	Domínio	Gerencial/Escalonamento	Estático
	Ocupação de CPU	Gerencial/Escalonamento	Dinâmico
	Ocupação de Memória	Gerencial/Escalonamento	Dinâmico

softwares configurados no servidor e obtidos através de métodos de *Web Services* no momento que o sensor é inicializado.

Algumas características do hardware da máquina serão monitoradas com detalhes, por exemplo, memória, onde será indicado o total e o que tem em cada *slot*, nome da estação, usuário logado, Sistema Operacional instalado, data de instalação.

Outra informação é quando foi feito o último *boot*, para estas informações mais estáticas será feito somente uma coleta pois estes dados para mudarem será preciso desligar a máquina para alterá-los.

Para processador, a informação coletada é de qual o *clock* máximo, o tipo do processador, versão. Para discos lógicos terá a informação se a máquina possui *Drive* de disquete ou se possui algum *drive* de CD ou DVD, e espaço livre do HD, o sistema de arquivos utilizado pelo dispositivo.

Dispositivos de rede fornecerá informações como o nome do dispositivo, o domínio que está conectado, o endereço *MAC address* e o endereço IP.

Estas informações serão coletadas com uma freqüência maior por alterarem com uma maior freqüência como por exemplo usuário logado e a medição do tempo de utilização de softwares.

3.4 Modelagem

O modelo de monitoramento pode ser considerado como composto por um conjunto de dados coletados e por um conjunto de elementos que manipulam esses dados. Por isso, a modelagem apresentada nesta seção apresenta tanto os dados coletados quanto a coleta. Como visto na Seção 3.2, existem três elementos responsáveis por

realizar a coleta: Sensor, Coletor local e Coletor global.

O sensor faz a coleta dos dados localmente na máquina, após é feito o processamento antes do envio das informações ao coletor local. É instalado localmente na máquina e inicializado no momento em que esta é ligada. Para descentralizar o processamento das informação a fim de diminuir o consumo do servidor, diminuindo o volume de informação a ser processada.

A Figura 3.2 mostra o diagrama de caso de uso, com a visão do sensor e todas as ações que serão tomadas por ele, cada elipse caracteriza uma funcionalidade no momento da coleta das informações da máquina, estas ações são gerenciadas pelo sensor onde ele decide qual funcionalidade será inicializada e executada, como buscar dados dos dispositivos de rede instalados na máquina, obter dados da memória, programas instalados, dados do Sistema Operacional e dos dispositivos lógicos.

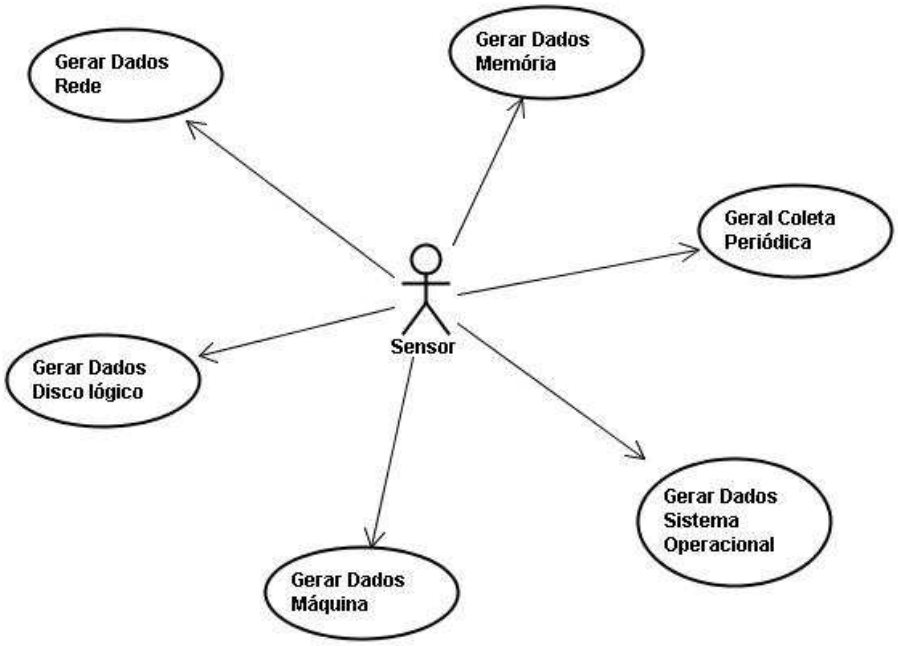


Figura 3.2: Caso de Uso: visão sistema protótipo

Coletor local é aquele que centraliza os dados do nó da grade.

Coletor global é responsável por fornecer uma visão aproximada de todos os recursos da grade. Para isso, ele deve receber informações dos coletores locais. Um coletor global pode receber os dados dos coletores locais de duas formas:

1. Pelo recebimento das informações em formato XML;

2. Por meio de replicação dos dados da base.

Isso obriga a que a máquina do coletor global possua tanto um bom processador e boa quantidade de memória, quanto tenha um servidor de banco de dados instalado.

No primeiro caso, o envio de dados no formato de um documento XML traz maior flexibilidade ao permitir que os dados coletados sejam armazenados até mesmo em arquivos texto comuns.

Porém, será necessário que o serviço do coletor global seja capaz de detectar a inclusão de novos arquivos com dados XML. Também deve tratar a integração de novos dados coletados com os antigos, consolidadando em um único documento XML, o que pode ser uma operação cara do ponto de vista de tempo processamento.

No segundo caso, o coletor local deve possuir uma base local onde as informações são armazenadas e, através do sistema de gerenciamento de bancos de dados, um processo de replicação é disparado. Para que isso seja possível, é necessário que a máquina do coletor global seja capaz de receber essas requisições, que são pesadas, sem prejudicar o desempenho da máquina.

No entanto, o uso de um SGBD (Sistema gerenciador de Banco de Dados) normalmente traz vantagens como maior segurança no acesso aos dados por exigir autenticação e autorização.

A Figura 3.3 contém o diagrama de sequência do protótipo, onde ilustra os passos para obtenção dos dados pelo sensor antes de enviá-los para o coletor local, desde o momento em que a máquina é ligada, passando pela coleta dos dados dinâmicos até o momento em que é feito o envio das informações para o coletor local. Antes de iniciar a coleta das informações o sensor verifica se as informações foram enviadas, caso não tenham sido enviada o sensor as envia para o coletor local.

Após o envio é feita a inicialização da classe responsável pela coleta inicial, completa com todas as informações.

Depois disto o sensor é avisado no momento de uma coleta dos dados dinâmicos, para saber se teve alteração ou atualização. A classe computador é responsável pelo armazenamento e atualização dos dados coletados.

No final do processo são enviados os dados para o coletor local.

Para os próximos envios é calculado o tempo com base em seu identificador, para que se diminua a incidência de máquinas enviando ao mesmo tempo, com o tempo calculado é feito a coleta dos dados dinâmicos, e por último entes de desligar é feito o envio das últimas informações.

A Figura 3.4 mostra o diagrama de classes do protótipo, onde temos todas as classes responsáveis pela obtenção e armazenamento dos dados. A modelagem foi baseada na modelagem feita pelo Buyya no projeto (Luther et al., 2005). Os diagramas foram feitos através da ferramenta Jude Community versão 3.0.

A classe *computador* é responsável pela obtenção dos dados estáticos ou que são alterados com uma menor freqüência, como: O identificador único onde cada máquina tem o seu, o protótipo pode coletá-lo via software.

- *Patrimonio*: identificador da máquina;

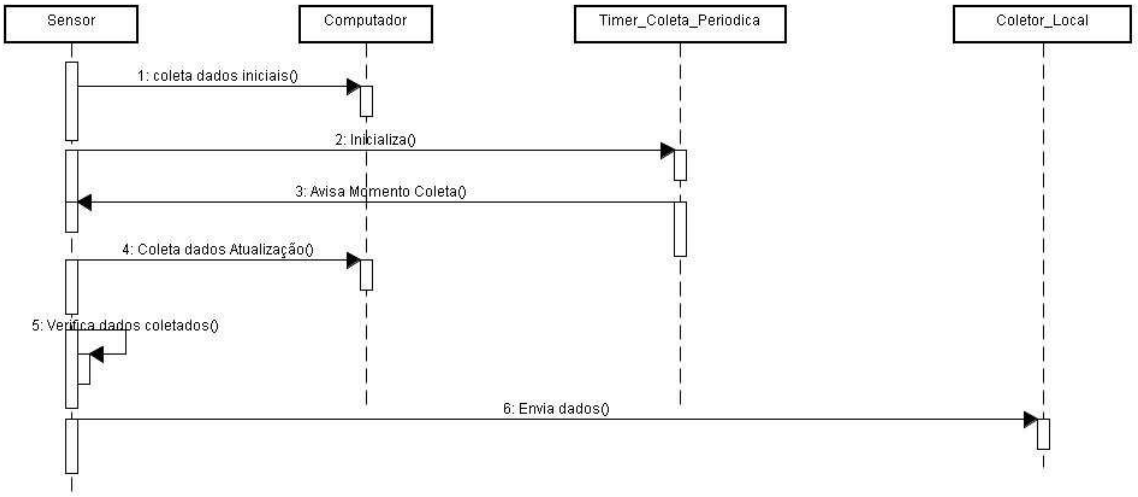


Figura 3.3: Diagrama de Sequência: visão sistema protótipo

- *Serie*: número de série da máquina;
- *Modelo*: modelo da máquina;
- *NomeComputador*: nome do computador;
- *Usuario*: último usuário logado na máquina;
- *SistemaOperacional*: nome do Sistema Operacional instalado na máquina;
- *UltimoBoot*: data que foi feito o último *boot* na máquina;
- *DataInstalacao*: data de instalação do Sistema Operacional.

A classe *Processadores* é uma estrutura que contém uma coleção da classe *Processador* que tem os seguintes atributos;

O *CpuId* o identificador que o Sistema Operacional atribui para o processador;

- *Patrimonio*: identificador da máquina;
- *CpuId*: identificador da CPU;
- *Caption*: *label* dado ao processador;
- *ClockMaximo*: velocidade máxima em Hz do processador;
- *Nome*: nome do processador;
- *Versao*: versão do processador;

A classe *Memorias* como a *Processadores*, também é uma estrutura da coleção da classe *Memoria* e os seu atributos são:

- *Patrimonio*: identificador da máquina;
- *NomeSlot*: nome dados ao slot;
- *Capacidade*: capacidade em *bytes* que o *paint* de memória suporta;
- *Localizacao*: em qual o slot está o *paint* de memória;
- *Tipomemoria*: código do tipo da memória;
- *Nome*: nome da memória;
- *Velocidade*: velocidade da memória;
- *Tag*: *label* onde consta o nome e a localização da memória.

A classe *Discos* é uma estrutura da classe *Discologico* cujo os atributos são:

- *Patrimonio* o identificador da máquina;
- *TipoDrive* indica o código do *Drive*;
- *Letra* indica a letra da unidade;
- *Descricao* indica a descrição do tipo de *drive*;
- *SistemaArquivos* indica o sistema de arquivos utilizado pelo *drive*;
- *Tamanho* indica o tamanho em *bytes* do *drive*;
- *EspacoLivre* indica a quantidade de espaço livre em *bytes*.

A classe *DispositivosRede* é uma coleção da classe *Dispositivorede* e seus atributos são:

- *Patrimonio*: identificador da máquina;
- *DeviceId*: identicador do dispositivo;
- *Descricao*: Descrição do dispositivo, como nome da placa de rede;
- *DNS*: os endereços DNS;
- *DominioDNS*: os servidores de DNS;
- *SufixoDNS*: os sufixos de DNS;

- *Hostname*: nome do Computador;
- *MacAddress*: número do MAC;
- *IP* indica o endereço IP.

A classe *Programas* é uma estrutura que contém a coleção da Classe *Programa* com os seguintes atributos:

- *Patrimonio*: identificador da máquina;
- *Executavel*: nome do executável;
- *local*: local (*Path*) onde foi instalado o programa;
- *DataInstalacao*: data de instalação do programa;
- *versao*: versão do programa instalado.

3.5 Protótipo

O protótipo foi desenvolvido em C#, o sensor implementado como um serviço instalado localmente no nó. A partir do momento que a máquina é ligada o sensor passará a monitorar o equipamento, para esta monitoração terá alguns atributos que serão responsáveis por pontos específicos da monitoração, como a medição do uso de software ou quantidade de memória.

O protótipo tem dois atributos do tipo TIMER (objeto que executa determinada ação em um intervalo pré definido) para coleta das informações na inicialização da classe.

Inicia-se as propriedades do serviço Windows e utilizando WMI (*Windows Management Instrumentation*) (WMI, 2006). Pega-se o nome da máquina e com base no nome busca-se o tempo para a inicialização de um dos *Timers*, somente depois deste tempo que será feita a primeira coleta das características e um posterior envio para o servidor através do *Web Service*.

O tempo é calculado através dos dois últimos algarismos do nome da máquina. No primeiro envio das informações é feita uma transferência total dos dados coletados, antes do próximo envio o sensor analisa as informações atuais e antes de enviar compara com as informações anteriores e somente após esta comparação é efetuada transferência das informações novas ou alteradas evitando o trafego de informações desnecessárias na rede.

Um problema encontrado foi a questão de um identificador único para as máquinas, dois atributos que poderiam ser usados são o número IP ou o número MAC.

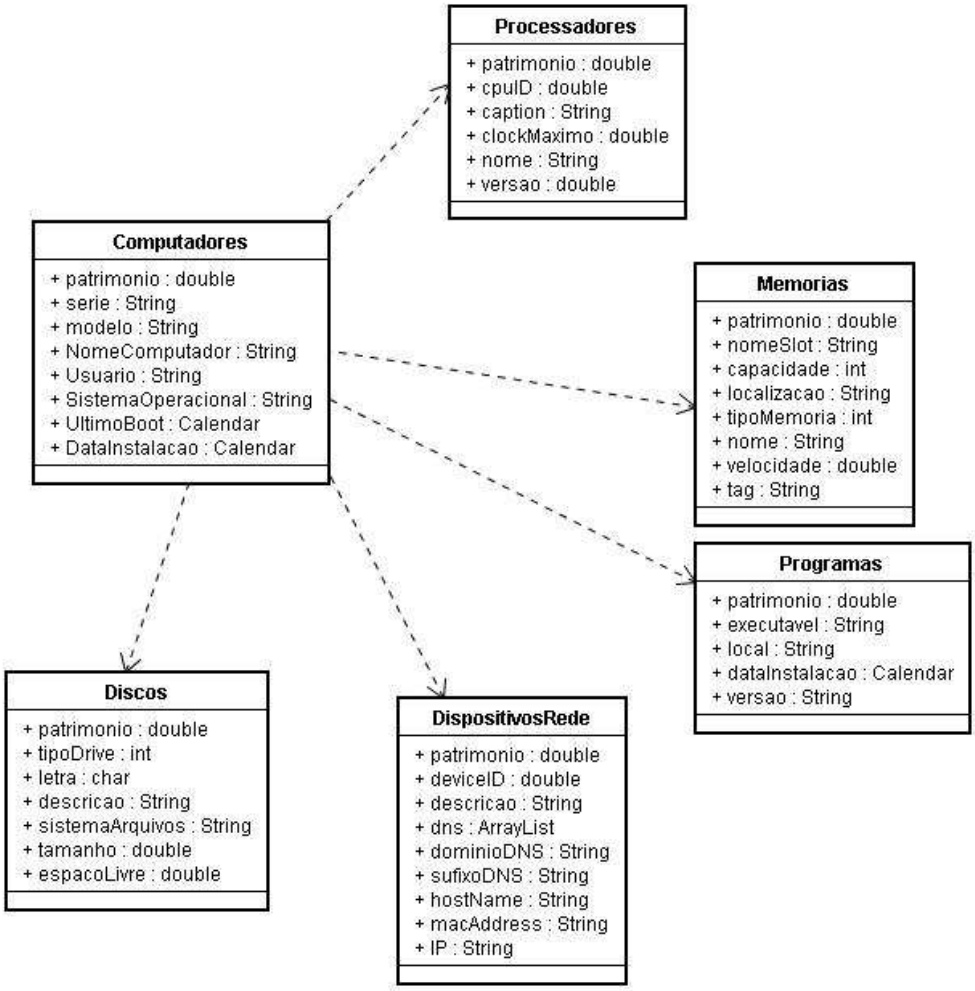


Figura 3.4: Diagrama de Classe para os dados

IP não é uma solução segura pois podem existir como no Banrisul, NAT, ou como na UFRJ (Universidade Federal do Rio de Janeiro), domínios que utilizam DHCP (*Dynamic Host Configuration Protocol*) , e número MAC podem repetir, não devem, então como solução para o ambiente corporativo Banrisul, foi utilizado o número de patrimônio dos equipamentos por ser um número único e por esta informação estar contida na BIOS.

O segundo *Timer* é inicializado depois da inicialização da máquina, ele é responsável por capturar as informações que não estáticas que serão enviadas ao servidor ao desligar o micro e haverá uma verificação ao ligar, caso não tenha sido enviado, será enviado ao ligar.

O segundo também será utilizado para coletar informações do nível de utilização da CPU, onde definimos que uma CPU ociosa está entre o intervalo de 0 a 29%, uma CPU ocupada está entre 30 e 79% e CPU sobrecarregada acima de 80%. O envio destas informações não será por um determinado período de tempo e sim transição de estado.

Os dados são armazenados em arquivos XML para que no momento do envio eles possam ser comparados com os atuais antes do envio, é feito a comparação e atualizados o arquivo XML com as alterações e mantendo sempre atualizados com as últimas informações, a Figura 3.5 mostra como fica o arquivo com as informações. O Anexo A contém o documento XML completo que foi obtido em nossos experimentos.

O Anexo B contém o documento XML gerado pelo sistema para envio para o coletor local das alterações obtidas na máquina.

```
<?xml version="1.0" encoding="utf-8" ?>
- <Computador xmlns:BMS="http://n061/bms/link">
    <Patrimonio>421557</Patrimonio>
    <Serie>32Q9N81</Serie>
    <Fabricante>Dell Inc.</Fabricante>
    <NomeComputador>TM0421557</NomeComputador>
    <Usuario>TM0421557\Administrador</Usuario>
    <SistemaOperacional>Microsoft Windows XP Professional</SistemaOperacional>
    <SistemaOperacionalSP>Service Pack 2</SistemaOperacionalSP>
    <UltimoBoot>2007-02-25T23:05:56.3750000-03:00</UltimoBoot>
    <DataInstalacao>2006-10-06T13:23:50.0000000-03:00</DataInstalacao>
  + <Processadores>
  + <Memorias>
  + <DiscosLogicos>
  + <DispositivosRede>
  + <Programas>
  </Computador>
```

Figura 3.5: Exemplo de arquivo XML Descrevendo um Computador

3.6 Viabilidade de porte para ambiente Linux

O Mono (Mono, 2006) é uma plataforma de desenvolvimento de código aberto baseada no framework .NET que permite aos desenvolvedores construir aplicações Linux e multi-plataforma.

A implementação do .NET do Mono é baseada nos padrões ECMA (Runtime Comum de Linguagens, ou CLR) para a C# e para a CLI (Infrastrutura Comum de Linguagens).

Patrocinado pela Novell, o Mono inclui ferramentas de desenvolvimento e a infraestrutura necessária pra executar aplicações cliente e servidor .NET.

O Mono inclui um compilador para a linguagem C#, um *engine* de *runtime* compatível com a ECMA, e as bibliotecas de classes. As bibliotecas incluem as bibliotecas de compatibilidade com o .NET da Microsoft (incluindo a ADO.NET e ASP.NET, as bibliotecas Mono e bibliotecas de terceiros).

Gtk#, como uma parte da ligação do .NET com o toolkit gtk+ e algumas bibliotecas GNOME podem ser encontradas neste último. Esta biblioteca permite você a construir aplicações Gnome totalmente nativas usando Mono e inclui o suporte para as interfaces de usuário. Além disso, os executáveis do Mono podem ser embarcados em aplicações para se obterempacotamento e distribuição mais fáceis. Além disto, o projeto Mono oferece uma IDE, um *debugger*, e um *browser* de documentação. Para o porte para linux será necessário o desenvolvimento de uma classe para obtenção dos dados do /proc das distribuições, porque a classe Management, responsável pela obtenção dos dados tanto hardware quanto software, não foi implementada no Mono sendo necessário a implementação. O /proc é um local onde o Kernel linux de todas as distribuições mantém um padrão.

Uma limitação para o Mono é o fato dele funcionar em distribuições baseadas no Gtk (the GIMP Toolkit) e não funcionar em outrar baseadas em GDM (*GNOME Display Manager*). Algumas distribuições foram testadas para montagem do ambiente para o porte como o Red Hat e Fedora.

3.7 Considerações finais

O modelo foi projetado para ser pouco intrusivo e pelo fato do envio das informações se feito somente com alterações ou atualizações não sobrecarregando a rede e também pela utilização de *Web Service* para a comunicação com o servidor central ele pode se comunicar com qualquer servidor central.

No próximo capítulo estão dados tentando mostrar a validade desta hipótese.

4 DADOS EXPERIMENTAIS

Este capítulo apresenta os resultados dos testes efetuados para tentar validar a hipótese de baixa intrusão e envio eficiente dos dados.

Os dados dos experimentos foram obtidos através de testes tanto nos ambientes de teste quanto de produção do Banrisul. Os experimentos apresentam-se desta forma: na Seção 4.1 analisa-se a intrusão do sensor na máquina sem se preocupar com dados da rede, na Seção 4.2 apresenta-se o levantamento da intrusão na rede. Finalmente na Seção 4.3 mostra o impacto da coleta, onde analisa-se a intrusão no servidor no momento do processamento das informações.

Para a média foi calculada a média aritmética simples e para o desvio foi calculado através da fórmula do desvio padrão (não-polarizado ou n-1) escrita assim:

$$\sqrt{\frac{n\sum x^2-(\sum x)^2}{n(n-1)}}$$

Esta é a fórmula comumente utilizada, estando disponível em ferramentas como MS Excel.

Os equipamentos utilizados para fazer estes experimentos tem a mesma configuração sendo todas Pentium 4 com 512 MB de memória, para cada experimento, varioi as aplicações que estão executando, e o horário da coleta dos dados.

4.1 Intrusão na Máquina

Neste experimento é tratado a questão da intrusão em máquinas sem se preocupar com o tempo que levará para envio das informações. Neste experimento o objetivo é provar que o protótipo não interfere nas atividades do usuário, o percentual sobe no momento da requisição mas para o processamento das informações ele baixa logo em seguida.

Os dados foram coletados através da execução do sensor, o tempo levado para a coleta completa das informações foi entre 8 e 15 segundos, e para uma coleta periódica de dados o tempo ficou entre 3 e 5 segundos.

A Tabela 4.1 mostra o acompanhamento do que ocorre durante a execução do sensor, monitorando a hipóteses de que sensores locais tem baixa intrusão e de que a estrutura hierárquica aliada ao envio apenas de informações relevantes causa pouco impacto nas máquinas e na rede durante o envio dos dados. Pelo fato do envio das informações com os dados completos do computador ser feito somente na primeira vez, as demais as informações são processadas e feita uma comparação com os dados anteriores e somente após análise é feito o envio com os dados alterados ou atualizados.

Na Tabela 4.1, a primeira coluna tem o equipamento que está sendo executado o sensor, na segunda tem a ocupação da CPU, trazendo os percentuais de ocupação, média e desvio padrão. Na terceira coluna tem quanto foi utilizado de memória pelo sensor e na última tem o tamanho do arquivo XML gerado com as informações do computador.

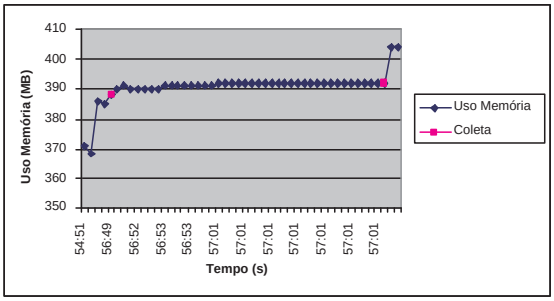
Tabela 4.1: Intrusão nas Máquinas

Equipamento	**Ocupação CPU**			**Memória**	**Tam. Arq. Gerado**
	Máx	Méd	Desvio		
Equip 1	50 %	24 %	19 %	3 MB	17 Kbytes
Equip 2	50 %	24 %	19 %	5 MB	15 Kbytes
Equip 3	49 %	23 %	16 %	4 MB	17 Kbytes
Equip 4	50 %	22 %	17 %	5 MB	16 Kbytes
Equip 5	50 %	21 %	15 %	6 MB	16 Kbytes
Equip 6	49 %	23 %	16 %	5 MB	17 Kbytes
Equip 7	50 %	24 %	19 %	7 MB	15 Kbytes
Equip 8	50 %	24 %	19 %	8 MB	16 Kbytes
Equip 9	50 %	24 %	19 %	8 MB	17 Kbytes
Equip 10	49 %	23 %	16 %	5 MB	16 Kbytes
Média	49,7 %	23,2 %	17,5 %	5,6 MB	16,2 Kbytes
Desvio Padrão	0,48 %	1,03 %	1,64 %	1,64 MB	0,78 Kbytes

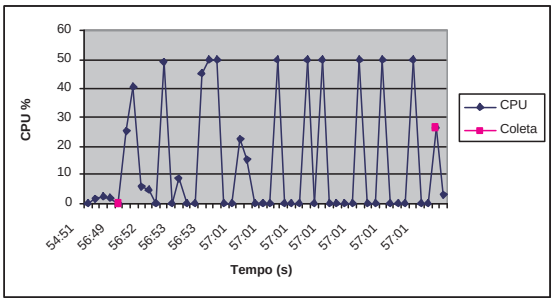
Tanto a Figura 4.1a quanto a Figura 4.1b mostram a intrusão do sensor no momento da execução da coleta dos dados, a diferença é que a Figura 4.1 faz um gráfico da utilização de memória e percentual de CPU em um horário que não tem utilização muito grande. A Figura 4.2 faz o gráfico de utilização de memória e percentual de CPU em um horário que a a utilização da rede e da máquina é grande.

A coleta seletiva é efetuada após a coleta completa, a primeira coleta é feita de todos os dados estáticos e dinâmicos, as demais são efetuadas somente para os dados dinâmicos.

A Tabela 4.2 ilustra a comparação de uma coleta total com uma coleta de dados estáticos. Na primeira coluna tem o equipamento onde foi feito o experimento.Na segunda tem o tempo da coleta total. Na terceira o tempo de uma coleta estática.



(a) Sensor Uso Memória

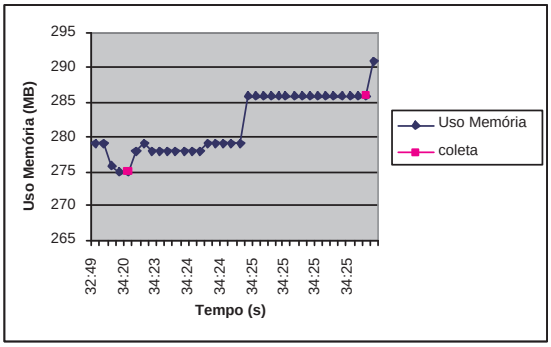


(b) Sensor Uso CPU

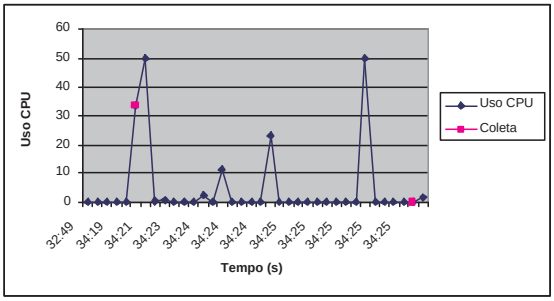
Figura 4.1: Acompanhamento Sensor em horário de pouca utilização

Tabela 4.2: Comparação coleta completa e seletiva

Equipamento	**Tempo Coleta Total**	**Tempo Coleta dados Estáticos**
Equip 1	8 s	3 s
Equip 2	5 s	3 s
Equip 3	5 s	3 s
Equip 4	7 s	4 s
Equip 5	8 s	3 s
Equip 6	6 s	3 s
Equip 7	7 s	4 s
Equip 8	6 s	4 s
Equip 9	8 s	4 s
Equip 10	7 s	3 s
Média	6,7 s	3,4 s
Desvio Padrão	1,15 s	0,51 s



(a) Sensor Uso Memória



(b) Sensor Uso CPU

Figura 4.2: Acompanhamento Sensor em horário de muita utilização

4.2 Intrusão na Rede

Este experimento visa demonstrar a intrusão do sensor na rede no momento do envio dos pacotes com os dados coletados.

A Tabela 4.3 mostra o acompanhamento da intrusão no momento do envio, medindo o tempo de envio e o tamanho dos pacotes enviados ao servidor. Na primeira coluna tem o Equipamento sem descrição uma vez que todos eles tem a mesma configuração, na segunda tem o tipo de pacote, na terceira o tempo de envio e na última o tamanho do pacote enviado. Todos os testes foram feitos com uma rede com a mesma largura de banda.

O equipamento marcado com um (*) indica que foi executado em um horário diferenciado ou seja em um horário em que a prioridade de execução e portas se dá a atualização de registros e processamento de grande porte.

Este horário de envio impactou no tempo levado para transferência do pacote pois neste período as portas de HTTP (HyperText Transfer Protocol) não tem prioridade.

4.3 Impacto da coleta independente de dados

O experimento visa medir o impacto da coleta independente de dados pelo sensor, executando em uma rede com conexão de 64 Kbps, utilizada em produção e com uma execução de 53 Kbps (Vercoza, 2006) normalmente da banda.

Tabela 4.3: Intrusão na Rede

Equipamento	Tipo Pacote	Temp. Envio	Tam. Pacote
Equip 1	Inicial	7 s	11 Kbytes
	Periódica	5 s	9 Kbytes
Equip 2	Inicial	8 s	12 Kbytes
	Periódica	5 s	10 Kbytes
Equip 3 (*)	Inicial	40 s	11 Kbytes
	Periódica	25 s	9 Kbytes
Equip 4	Periódica	20 s	9 Kbytes
	Periódica	15 s	9 Kbytes
Equip 5	Inicial	8 s	12 Kbytes
	Periódica	6 s	10 Kbytes
Equip 6	Inicial	9 s	11 Kbytes
	Periódica	4 s	9 Kbytes
Equip 7	Inicial	9 s	11 Kbytes
	Periódica	6 s	9 Kbytes
Equip 8	Inicial	7 s	12 Kbytes
	Periódica	5 s	10 Kbytes
Equip 9	Periódica	6 s	9 Kbytes
	Periódica	7 s	9 Kbytes
Equip 10	Inicial	10 s	11 Kbytes
	Periódica	8 s	9 Kbytes
Média	Inicial	12,4 s	10,9 Kbytes
	Periódica	8,6 s	9,3 Kbytes

O envio em tempos distintos significa o envio das informações em tempos diferentes, sem se preocupar com a sobrecarga do servidor, medindo somente a taxa de utilização da banda e o tempo gasto para o envio.

Envio ao mesmo tempo foi o envio de todos os dados ao mesmo tempo e monitorando o custo do processamento no servidor para o processamento das informações recebidas cos sensores.

4.3.1 Envio em tempos Distintos

O experimento analisará os dados para o envio das informações em tempos distintos.

A Tabela 4.4 mostra o acompanhamento da taxa de ocupação da banda pelo envio das informações pelo sensor, na primeira coluna tem os equipamentos utilizados (todos com a mesma configuração), na segunda tem o tempo levado para o envio, e a terceira tem a taxa de ocupação da banda com o envio das informações.

4.3.2 Envio ao mesmo tempo

O experimento analisará o tempo e a intrusão no servidor no momento do recebimento dos dados.

A Tabela 4.5 mostra o tempo gasto para efetuar o envio dos dados ao servidor, sendo que todos os equipamentos enviaram ao mesmo tempo, em uma grade de teste e com um *link* de 64 Kbps, a taxa de ocupação da rede ficou em torno de 8,5%. No servidor o uso de CPU chegou a 7% no momento do processamento do *Web Service*

Tabela 4.4: Envio dos dados em Tempos Distintos

Equipamento	**Temp. Envio**	**Ocupação Banda**
Equip 1	7 s	2 %
Equip 2	8 s	1,8 %
Equip 3	9 s	2,1 %
Equip 4	7 s	2 %
Equip 5	9 s	2,3 %
Equip 6	7 s	1,8 %
Equip 7	8 s	2 %
Equip 8	8 s	1,9 %
Equip 9	9 s	2 %
Equip 10	8 s	2 %
Média	8 s	1,99 %
Desvio Padrão	0,81 s	0,14 %

Tabela 4.5: Envio dos dados ao mesmo tempo

Equipamento	**Temp. Envio**
Equip 1	7 s
Equip 2	6 s
Equip 3	7 s
Equip 4	7 s
Equip 5	7 s
Equip 6	8 s
Equip 7	8 s
Equip 8	6 s
Equip 9	6 s
Equip 10	7 s
Média	6,9 s
Desvio Padrão	0,74 s

5 CONCLUSÃO

Neste trabalho apresentamos os principais aspectos que caracterizam uma grade computacional e a importância do serviço de monitoramento de recursos para auxílio a ferramentas escalonadoras e administradores de redes. Dentro deste contexto apresentamos um novo modelo de monitoramento, que permite obter informações de hardware e software, com o envio controlado para evitar sobrecarga da rede.

O sensor apresentou uma baixa taxa de intrusão e uma baixa sobrecarga da rede para envio dos dados.

Conforme os dados experimentais obtidos, um dos motivos do seu desempenho deve-se ao fato de guardar localmente as informações dos dados anteriores para que no momento em que o sensor for enviar novamente as informações para o coletor local seja feita uma comparação para determinar as diferenças existentes entre o último envio.

Posteriormente é feito uma análise das informações obtidas no momento da execução do sensor, e com base nisto é enviado para o *Web Service* somente as diferenças. Desta forma nã há sobrecarga da rede com estas informações.

No primeiro envio que será o que contém a maior quantidade de dados onde a sobrecarga na rede é maior.

O sensor com base em um identificador calcula o tempo para o primeiro envio, diminuindo a probabilidade de que muitas máquinas enviem ao mesmo tempo sobrecarregando a rede com estes dados.

Proporcionar informações de software e algo que nem todas as ferramentas existentes hoje no ambiente de grade proporcionam.

O modelo proporciona não só informações de software como também a localização (path), a data de instalação e a versão do software.

Algumas destas informações só servem para administradores de rede, mas outras auxiliam ferramentas de escalonamento pelo fato de poder identificar a localização de determinado software. Além de saber dos software existentes na grade também

saber em qual máquina ele está instalado, as características da máquina, auxiliando a fazer um melhor escalonamento.

Alguns aspectos do modelo não estão sendo tratados neste momento por exemplo:

A interface para visualização dos dados monitorados, onde fornecerá relatórios buscando por características ou que o usuário faça sua própria *query* para uma consulta customizada.

A implementação da base de dados, *scripts* de criação da base e definição de qual SGBD (Sistema Gerenciador de Banco de Dados) .

além disso, conforme apresentado na Seção 3.6, uma das atividades futuras é o porte do protótipo para o ambiente Linux utilizando o framework Mono e o desenvolvimento da classe para obtenção dos dados através do /proc que obtém informações do *Kernel* que é padrão em todas as distribuições.

Finalmente, um trabalho importante é a avaliação deste modelo em um ambiente de grade heterogêneo, tal como os tipicamente existentes em instituições de pesquisa.

APÊNDICE A DOCUMENTO XML DO SENSOR COMPLETO

```
 <?xml version="1.0" encoding="utf-8" ?>
<Computador xmlns:BMS="http://n061/bms/link">
 <Patrimonio>421557</Patrimonio>
 <Serie>32Q9N81</Serie>
 <Fabricante>Dell Inc.</Fabricante>
 <NomeComputador>TM0421557</NomeComputador>
 <Usuario>TM0421557\Administrador</Usuario>
 <SistemaOperacional>Microsoft Windows XP Professional</SistemaOperacional>
 <SistemaOperacionalSP>Service Pack 2</SistemaOperacionalSP>
 <UltimoBoot>2007-02-25T23:05:56.3750000-03:00</UltimoBoot>
 <DataInstalacao>2006-10-06T13:23:50.0000000-03:00</DataInstalacao>
<Processadores>
<Processador>
 <BMS:Patrimonio>421557</BMS:Patrimonio>
 <BMS:CPUId>CPU0</BMS:CPUId>
 <BMS:Caption>x86 Family 15 Model 4 Stepping 3</BMS:Caption>
 <BMS:ClockMaximo>3192</BMS:ClockMaximo>
 <BMS:Nome>Intel(R) Pentium(R) 4 CPU 3.20GHz</BMS:Nome>
 <BMS:Versao>Modelo 4, Nível 3</BMS:Versao>
 </Processador>
<Processador>
 <BMS:Patrimonio>421557</BMS:Patrimonio>
 <BMS:CPUId>CPU1</BMS:CPUId>
 <BMS:Caption>x86 Family 15 Model 4 Stepping 3</BMS:Caption>
 <BMS:ClockMaximo>3192</BMS:ClockMaximo>
 <BMS:Nome>Intel(R) Pentium(R) 4 CPU 3.20GHz</BMS:Nome>
 <BMS:Versao>Modelo 4, Nível 3</BMS:Versao>
 </Processador>
 </Processadores>
<Memorias>
<Memoria>
 <BMS:Patrimonio>421557</BMS:Patrimonio>
 <BMS:NomeSlot />
 <BMS:Capacidade>268435456</BMS:Capacidade>
 <BMS:Localizacao>DIMM_1</BMS:Localizacao>
 <BMS:TipoMemoria>0</BMS:TipoMemoria>
 <BMS:Nome>Memória física</BMS:Nome>
 <BMS:Velocidade>533</BMS:Velocidade>
 <BMS:Tag>Physical Memory 0</BMS:Tag>
 </Memoria>
<Memoria>
```

```
 <BMS:Patrimonio>421557</BMS:Patrimonio>
 <BMS:NomeSlot />
 <BMS:Capacidade>268435456</BMS:Capacidade>
 <BMS:Localizacao>DIMM_2</BMS:Localizacao>
 <BMS:TipoMemoria>0</BMS:TipoMemoria>
 <BMS:Nome>Memória física</BMS:Nome>
 <BMS:Velocidade>533</BMS:Velocidade>
 <BMS:Tag>Physical Memory 2</BMS:Tag>
 </Memoria>
 </Memorias>
<DiscosLogicos>
<DiscoLogico>
 <BMS:Patrimonio>421557</BMS:Patrimonio>
 <BMS:TipoDrive>2</BMS:TipoDrive>
 <BMS:Letra>A:</BMS:Letra>
 <BMS:Descricao>Unidade de disquete de 3 1/2 polegadas</BMS:Descricao>
 <BMS:SistemaArquivos />
 <BMS:Tamanho>0</BMS:Tamanho>
 <BMS:EspacoLivre>0</BMS:EspacoLivre>
 </DiscoLogico>
<DiscoLogico>
 <BMS:Patrimonio>421557</BMS:Patrimonio>
 <BMS:TipoDrive>3</BMS:TipoDrive>
 <BMS:Letra>C:</BMS:Letra>
 <BMS:Descricao>Disco fixo local</BMS:Descricao>
 <BMS:SistemaArquivos>NTFS</BMS:SistemaArquivos>
 <BMS:Tamanho>39991275520</BMS:Tamanho>
 <BMS:EspacoLivre>36686491648</BMS:EspacoLivre>
 </DiscoLogico>
<DiscoLogico>
 <BMS:Patrimonio>421557</BMS:Patrimonio>
 <BMS:TipoDrive>5</BMS:TipoDrive>
 <BMS:Letra>D:</BMS:Letra>
 <BMS:Descricao>Disco CD-ROM</BMS:Descricao>
 <BMS:SistemaArquivos />
 <BMS:Tamanho>0</BMS:Tamanho>
 <BMS:EspacoLivre>0</BMS:EspacoLivre>
 </DiscoLogico>
<DiscoLogico>
 <BMS:Patrimonio>421557</BMS:Patrimonio>
 <BMS:TipoDrive>2</BMS:TipoDrive>
 <BMS:Letra>E:</BMS:Letra>
 <BMS:Descricao>Disco removível</BMS:Descricao>
 <BMS:SistemaArquivos>FAT32</BMS:SistemaArquivos>
 <BMS:Tamanho>255881216</BMS:Tamanho>
 <BMS:EspacoLivre>35057664</BMS:EspacoLivre>
 </DiscoLogico>
 </DiscosLogicos>
<DispositivosRede>
<DispositivoRede>
 <BMS:Patrimonio>421557</BMS:Patrimonio>
 <BMS:DeviceID>7</BMS:DeviceID>
 <BMS:Descricao>Broadcom NetXtreme Gigabit Ethernet - Miniporta do agendador de pacotes</BMS:Descricao>
 <BMS:DominioDNS>LOCAL.AG0915.BERGS</BMS:DominioDNS>
 <BMS:HostName>TM0421557</BMS:HostName>
 <BMS:MACAddress>00:12:3F:FD:B4:5C</BMS:MACAddress>
 <BMS:IP>10.64.112.47</BMS:IP>
 <BMS:DNS />
 <BMS:SufixoDNS />
```

```
 </DispositivoRede>
 </DispositivosRede>
<Programas>
 <BMS:Patrimonio>421557</BMS:Patrimonio>
 <BMS:Nome>Microsoft .NET Framework 1.1 Brazilian Portuguese Language Pack</BMS:Nome>
 <BMS:Caminho />
 <BMS:DataInstalacao>2004-11-03T21:00:00.0000000-03:00</BMS:DataInstalacao>
 <BMS:Versao>1.1.4322</BMS:Versao>
 </Programa>
<Programa>
 <BMS:Patrimonio>421557</BMS:Patrimonio>
 <BMS:Nome>J2SE Runtime Environment 5.0 Update 4</BMS:Nome>
 <BMS:Caminho />
 <BMS:DataInstalacao>2006-10-05T21:00:00.0000000-03:00</BMS:DataInstalacao>
 <BMS:Versao>1.5.0.40</BMS:Versao>
 </Programa>
<Programa>
 <BMS:Patrimonio>421557</BMS:Patrimonio>
 <BMS:Nome>McAfee VirusScan Enterprise</BMS:Nome>
 <BMS:Caminho>C:\Arquivos de programas\Network Associates\VirusScan\</BMS:Caminho>
 <BMS:DataInstalacao>2006-10-05T21:00:00.0000000-03:00</BMS:DataInstalacao>
 <BMS:Versao>8.0.0</BMS:Versao>
 </Programa>
<Programa>
 <BMS:Patrimonio>421557</BMS:Patrimonio>
 <BMS:Nome>Microsoft .NET Framework 1.1</BMS:Nome>
 <BMS:Caminho />
 <BMS:DataInstalacao>2005-04-21T21:00:00.0000000-03:00</BMS:DataInstalacao>
 <BMS:Versao>1.1.4322</BMS:Versao>
 </Programa>
</Programas>
</Computador>
```

APÊNDICE B DOCUMENTO XML DO SENSOR ALTERAÇÃO

```
<?xml version="1.0" encoding="utf-8"?>
<Computador xmlns:BMS="http://n061/bms/link">
  <Patrimonio>421557</Patrimonio>
  <Serie>32Q9N81</Serie>
  <Fabricante>Dell Inc.</Fabricante>
  <NomeComputador>TM0421557</NomeComputador>
  <Usuario>TM0421557\Administrador</Usuario>
  <SistemaOperacional>Microsoft Windows XP Professional</SistemaOperacional>
  <SistemaOperacionalSP>Service Pack 2</SistemaOperacionalSP>
  <UltimoBoot>2007-02-25T23:05:56.3750000-03:00</UltimoBoot>
  <DataInstalacao>2006-10-06T13:23:50.0000000-03:00</DataInstalacao>
  <DiscosLogicos>
    <DiscoLogico>
      <BMS:Patrimonio>421557</BMS:Patrimonio>
      <BMS:TipoDrive>2</BMS:TipoDrive>
      <BMS:Letra>A:</BMS:Letra>
      <BMS:Descricao>Unidade de disquete de 3 1/2 polegadas</BMS:Descricao>
      <BMS:SistemaArquivos />
      <BMS:Tamanho>0</BMS:Tamanho>
      <BMS:EspacoLivre>0</BMS:EspacoLivre>
    </DiscoLogico>
    <DiscoLogico>
      <BMS:Patrimonio>421557</BMS:Patrimonio>
      <BMS:TipoDrive>3</BMS:TipoDrive>
      <BMS:Letra>C:</BMS:Letra>
      <BMS:Descricao>Disco fixo local</BMS:Descricao>
      <BMS:SistemaArquivos>NTFS</BMS:SistemaArquivos>
      <BMS:Tamanho>39991275520</BMS:Tamanho>
      <BMS:EspacoLivre>36686352384</BMS:EspacoLivre>
    </DiscoLogico>
    <DiscoLogico>
      <BMS:Patrimonio>421557</BMS:Patrimonio>
      <BMS:TipoDrive>5</BMS:TipoDrive>
      <BMS:Letra>D:</BMS:Letra>
      <BMS:Descricao>Disco CD-ROM</BMS:Descricao>
      <BMS:SistemaArquivos />
      <BMS:Tamanho>0</BMS:Tamanho>
      <BMS:EspacoLivre>0</BMS:EspacoLivre>
    </DiscoLogico>
  </DiscosLogicos>
```

```
</Computador>
```

REFERÊNCIAS

Andrade, N. (2006). *Condor*. http://www.lsd.ufcg.edu.br/

Bona, L. C. E. d. (2004). *HyperGrid: uma plataforma distribuída e confiável para computação em grade*. Teste de doutorado, Universidade Tecnológica Federal do Paraná, Paraná.

Chin, J., Coveney, P. V., and Jha, S. (2004). Grid computing: Making the global infrastructure a reality. *Computing in Science Engineering*, 6(5):15–77.

Dantas, M. (2005). *Computação Distribuída de Alto Desempenho*.

de Camargo, R. Y., Goldchleger, A., Carneiro, M., and Kon, F. (2004). *Grid: An Architectural Pattern*. Monticello,Illinois, USA.

de Mattos, É. C. T. (2003). *Análise e construção de um ambiente de grade computacional peer-to-peer com ênfase no balanciamento de carga*. Tese de doutorado, Universidade Federal de São Carlos, São Carlos.

de P. Nascimento, A., da C. Sena, A., Silva, J. A. D., Vianna, D. Q. C., Boeres, C., and Rebello, V. E. F. (2005). *Managing the Execution of Large Scale MPI Applications on Computational Grids*. SBAC - 2005.

dos Santos, L. A. S. (2006). *Uma proposta de escalonamento descentralizado de tarefas para Computação em Grade*. Tese de mestrado, Universidade Federal do Rio Grande do Sul, Porto Alegre.

Foster, I., Kessekman, C., and Tuecke, S. (2001). The anatomy of the grid: enabling scalable virtual organizations. *The International Journal of High Performance Computing Applications*, 15(3):200–222.

Foster, I. and Kesselman, C. (2003). *The Grid 2: Blueprint for a New Computing Infrastructure*.

Ganglia (2006). *Ganglia Monitoring System*. http://ganglia.sourceforge.net/.

Globus (2006). *Globus*. http://www.globus.org/.

GridBus (2006). *GridBus*. http://www.gridbus.org/.

GT3 (2006). *Globus Toolkit versão 3 (GT3)*. http://www.globus.org/toolkit/downloads/3.2.1/.

GT4 (2006). *Globus Toolkit versão 4 (GT4)*. http://www.globus.org/toolkit/downloads/4.1.1/.

Hawkeye (2006a). *Hawkeye*. http://www.cs.wisc.edu/condor/hawkeye/.

Hawkeye (2006b). *Hawkeye Software*. http://www.cs.wisc.edu/condor/cluster/hawkeye/.

Hollingsworth, J. and Tierney, B. (2004). The grid: Blueprint for a new computing infrastructure. *Morgn Kaufmann*, pages 319–352.

Jain, R. (1991). *The art of computer systems performance analysis*. John Wiley & Sons, USA.

Luther, A., Buyya, R., Ranjan, R., and Venugopal, S. (2005). Peer-to-peer grid computing and a .net-based alchemi framework. In *High Performance Computing: Paradigm and Infrastructure, Laurence Yang and Minyi Guo (eds)*, pages 403–429, New Jersey, USA. IEEE Computer Society.

Mangan, P. K. V. (2006). *Um Modelo de Gerenciamento Hierárquico de Aplicações em Ambiente de Computação em Grade*. Teste de doutorado, Universidade Federal do Rio de Janeiro, Rio de Janeiro.

Massie, M. L., Chun, B. N., and Culler, D. E. (2004). The ganglia distributed monitoring system:design, implementation, and experience.

MonaLisa (2006). *MonaLisa*. http://monalisa.caltech.edu/.

Mono (2006). *Projeto Mono Brasil*. http://monobrasil.softwarelivre.org.

NetLogger (2006). *NetLogger Toolkit*. http://www-didc.lbl.gov/NetLogger/.

Neves, M. V., Scheid, T., Schnorr, L. M., and Charão, A. S. (2004). *Integração de Ganglia, libRastro e Pajé para o Monitoramento de Aplicações Paralelas*. Sociedade Brasileira de Computação, Porto Alegre, RS, Brazil.

R-GMA (2006). *R-GMA*. http://www.r-gma.org/.

Remos (2006). *Remos: Resource Monitoring for Network-Aware Applications*. http://www.cs.cmu.edu/ cmcl/remulac/remos.html.

Vercoza, L. G. (2006). Dados da monitoração da rede do banrisul. Comunicação Pessoal.

WebMDS (2006). *WebMDS*. http://www.globus.org/toolkit/docs/4.0/info/webmds/.

WMI (2006). *WMI*. http://www.microsoft.com/whdc/system/pnppwr/wmi/default.mspx.

Wolski, R., Spring, N., and Hayes, J. (1999). The Network Weather Service: A distributed resource performance forecasting service for metacomputing. *Journal of Future Generation Computing Systems*, 15(5–6):757–768.