Integração do Gerenciamento de Clusters de Alto Desempenho com o Gerenciamento de Redes através de Solução baseada em SNMP

Rodrigo Sanger Alves[1], Clarissa Cassales Marquezan, Lisandro Zambenedetti Granville, Philippe Olivier Alexandre Navaux

Instituto de Informática – Universidade Federal do Rio Grande do Sul (UFRGS)
Caixa Postal 15.064 – 91.501-970 – Porto Alegre – RS – Brasil

{sanger,clarissa,granville,navaux}@inf.ufrgs.br

Abstract. *The management of high-performance clusters, in several situations, needs to be integrated with the management of the remaining of the computer network. In order to achieve such an integration, a network management architecture and protocol, such as the SNMP, can also be used for the management of clusters. This paper investigates the required management integration, through the implementation of SNMP agents deployed in the infrastructure of a cluster and, as well, also investigates the development of a management system for use over SNMP agents, themselves.*

Resumo. *O gerenciamento de clusters de alto desempenho necessita, em diversas situações, ser integrado ao gerenciamento do restante da rede de computadores. Para se obter essa integração pode-se adotar uma arquitetura e um protocolo padronizado de gerenciamento de redes, tal como o SNMP, para o gerenciamento de clusters. Este artigo investiga essa integração de gerenciamento, através da implementação de agentes SNMP, para emprego na infra-estrutura de clusters, bem como, também o desenvolvimento de um sistema para uso no gerenciamento destes mesmos agentes SNMP.*

1. Introdução

A pesquisa a respeito do uso de *clusters* (agregados) de computadores tem crescido consideravelmente nos últimos anos, principalmente quando *clusters* são utilizados como substitutos de supercomputadores em algumas aplicações que exigem processamento de alto desempenho [Buyya 1999]. A popularização dos *clusters*, nesse contexto, decorre, principalmente, do fato dos *clusters* serem uma alternativa de baixo custo e que proporciona uma expansão mais fácil do poder computacional através da implantação de mais nodos trabalhadores. A evolução e a aplicabilidade dos *clusters* também é conseqüência dos diversos *softwares* que foram desenvolvidos para a instalação, gerenciamento, programação e uso geral de *clusters*.

A complexidade de manutenção da infra-estrutura de nodos que formam um *cluster* é tipicamente proporcional ao próprio número de nodos de um *cluster*. Gerenciar um *cluster* sem o uso de ferramentas que automatizem processos pode ser uma tarefa praticamente inviável, principalmente quando o número de nodos envolvidos for

[1] Bolsista CNPq

grande. Assim, cada vez mais é necessária a existência de ferramentas que venham a auxiliar o administrador de um *cluster* a lidar com o número crescente de nodos.

Atualmente, existem à disposição diversas ferramentas que automatizam o gerenciamento de *clusters*, porém, a maioria delas tem sua própria linguagem e sua própria forma de funcionamento; não existe, portanto, uma padronização nos processos de gerenciamento de *clusters*. Esse problema de padronização dificulta a integração das ferramentas de gerenciamento de *clusters* com outras ferramentas de gerenciamento. Essa situação torna-se mais grave quando se leva em conta o fato de que os *clusters* estão inseridos na rede de computadores de uma instituição ou de uma empresa. Nesse caso, não é raro que o administrador da rede também venha a ser um administrador de *clusters*. Considerando as dificuldades de integração das ferramentas de gerência, o administrador do *cluster*/rede acaba sendo forçado a usar um conjunto de ferramentas para gerenciar a rede, e um outro conjunto de ferramentas para gerenciar os *clusters*.

Outro ponto a ser considerado é o fato de que os *clusters* são a base para a construção de *grids* computacionais de alto desempenho baseados no conceito de "*clusters* de *clusters*". Assim, pode-se considerar que, para realizar o gerenciamento de *grids*, um passo importante é o gerenciamento de *clusters*. O administrador do *cluster* pode ser, no caso de *grids*, novamente o próprio administrador da rede. Então, para um administrador de rede disposto a prover serviços de *grid* é importante que os dispositivos de sua rede e os *clusters* que vão compor o *grid* consigam interagir e possam ser gerenciados por uma ferramenta única ou, no mínimo, através de um ambiente integrado.

Assim, existe uma clara necessidade de se aproximar o gerenciamento de *clusters* do gerenciamento do restante da rede. Visto que a solução tradicional e amplamente aceita para o gerenciamento de redes TCP/IP é baseada no protocolo SNMP (*Simple Network Management Protocol*) [Case et. al. 1990], a integração com o gerenciamento de *clusters* pode ser conseguida através da introdução do suporte ao SNMP nas infra-estruturas dos *clusters*. Este artigo apresenta um sistema para gerenciamento de *clusters* baseado em SNMP usado no gerenciamento de dois *clusters* de nossos laboratórios. O sistema é composto por um agente SNMP e uma estação de gerenciamento baseada na Web. O acesso ao agente SNMP possibilita, por exemplo, a configuração de recursos internos ao *cluster*, tais como processos e alocação de CPUs.

O restante deste artigo está organizado da seguinte forma. A Seção 2 apresenta os trabalhos relacionados às ferramentas de gerenciamento de *clusters*. Na Seção 3 é apresentada a MIB (*Management Information Base*) definida para *clusters* e suportada pelo agente SNMP, além de detalhes de implementação do mesmo. A Seção 4 apresenta um estudo de caso com o sistema de gerenciamento baseado na Web e suas operações básicas. Por fim, as conclusões e os trabalhos futuros são descritos na Seção 5.

2. Gerenciamento de *Clusters* de Alto Desempenho

Os *clusters* surgiram da idéia de reunir o poder computacional de diversos computadores em uma única máquina virtual (um agregado de máquinas), trabalhando para a resolução de problemas através de técnicas de programação paralela e distribuída. O grande sucesso que os *clusters* atingiram está ligado ao fato de que, em algumas aplicações científicas (e.g. simulação), essas máquinas virtuais podem oferecer, mesmo com um custo baixo, desempenho próximo ao dos supercomputadores tradicionais.

Outro fator importante na popularização dos *clusters* é o fato destes poderem ser construídos a partir de equipamentos de *hardware* simples, encontrados em qualquer prateleira de loja de informática. Assim, os *clusters* são ditos máquinas construídas a partir de componentes COTS (*Commodity Off-The-Shelf*).

Segundo Buyya [Buyya 1999], um *cluster* é um tipo de sistema para processamento paralelo e distribuído que consiste de uma coleção de computadores *stand-alone* (nodos) interconectados e trabalhando juntos como um único recurso computacional integrado. Os componentes básicos de um *cluster* são: *front-end*, nodos e rede de interconexão. O ***front-end*** é a máquina que apresenta aos usuários do *cluster* a visão de um recurso computacional único e integrado. O papel do *front-end* é fundamental na concepção de um *cluster*, pois implementa o ponto único de acesso através do qual os usuários submetem e compilam aplicações a serem executadas no *cluster*. **Nodos** são, normalmente, computadores pessoais de uso geral com capacidade de interconexão à rede que compõe o *cluster*. Com alguma freqüência são utilizadas também máquinas multiprocessadas como nodos. Por fim, a **rede de interconexão** é a responsável por permitir a comunicação entre processos rodando em diferentes nodos. Existem diversas opções de tecnologias de interconexão à disposição para a montagem de *clusters*, sendo a escolha um compromisso entre o desempenho que será obtido e a quantia que se quer investir. Alguns exemplos de tecnologias de interconexão usadas em *cluster* são: Fast-Ethernet, Gigabit Ethernet, Myrinet e SCI (*Scalable Coherent Interface*) [Buyya 1999].

Atualmente, diferentemente do que acontece na área de gerenciamento de redes, não há uma noção bem definida e largamente aceita sobre o que é realmente o gerenciamento de *clusters*. De fato, diferentes conceitos são usados com o intuito de definir esse gerenciamento, mas tais conceitos são geralmente pouco precisos e até mesmo opostos, o que torna difícil seu entendimento. Dada essa situação, este artigo apresenta algumas ferramentas de gerenciamento de *clusters* divididas em três grupos principais: ferramentas para monitoração de *clusters*, ferramentas para alocação de nodos a usuários, e ferramentas administrativas para automação de tarefas. Como as funcionalidades encontradas nas ferramentas desses grupos são bastante próximas, não é raro que algumas ferramentas implementem funções de mais de um grupo.

As **ferramentas para monitoramento de *clusters*** são usadas para verificar o estado e a utilização dos recursos de um *cluster* (e.g. processadores, memória, e rede de interconexão). Algumas dessas ferramentas podem suportar, além da monitoração do *cluster*, tarefas para controle e configuração da infra-estrutura. O Ganglia [Massie 2003] é um dos sistemas para monitoração distribuída de *clusters* e *grids* mais difundidos atualmente. A escalabilidade do Ganglia permite a monitoração entre-*cluster* e intra-*cluster*. Ganglia utiliza XML (*Extensible Markup Language*) para representação de dados, XDR (*External Data Representation*) para transporte portável de dados, e RRDTool para armazenamento e visualização de informações. O sistema provê uma visualização gráfica dos recursos do *cluster* tais como processador e memória de cada nodo, usuários conectados e respectivos processos. Outra ferramenta é SIMONE (*SNMP-based Monitoring System for Network Computing*) [Subramanyan et. al. 2000]. SIMONE é uma das poucas ferramentas que possui uma integração com a arquitetura de gerenciamento SNMP, seguindo regras que permitem a sua associação com outras ferramentas já desenvolvidas para gerenciamento de redes. SIMONE fornece descrição das máquinas, informações sobre processos (e.g. tempo de CPU, memória usada),

informações sobre o nodo (e.g. porcentagem de uso de CPU, memória em uso, interfaces de comunicação) e informações sobre o tráfego de dados de cada nodo. Basicamente, SIMONE compreende as mesmas informações de gerenciamento disponibilizadas pelo Ganglia.

As **ferramentas para alocação de nodos a usuários** auxiliam no compartilhamento dos recursos do *cluster* entre vários usuários em um determinado momento. Essas ferramentas permitem a fragmentação lógica do *cluster* de forma que um usuário possa reservar um subconjunto dos nodos para a execução de uma aplicação, isolando assim a influência de outras aplicações na sua execução. Além dessa possibilidade de alocação de nodos, algumas ferramentas oferecem facilidades de escalonamento de tarefas com a utilização de alguma política de fila de execução (e.g. FIFO - *First In First Out* ou SJF - *Shortest Job First*). A ferramenta mais difundida nesse grupo é o PBS (*Portable Batch System*) [PBS 2004]. O PBS possibilita que os usuários aloquem nodos do *cluster* para acesso exclusivo e assim executem suas aplicações sem a interferência de outras aplicações. Para controlar o uso excessivo do *cluster* por parte de apenas um usuário, prejudicando os demais interessados, o PBS possibilita a implantação de políticas de uso para o *cluster* que arbitram a quantidade máxima de nodos e de horas que uma reserva pode conter.

Por fim, **as ferramentas administrativas para automação de tarefas** são compostas por ferramentas com funcionalidades (e.g. replicação de imagem de nodos e comandos paralelos) que não se enquadram nos dois grupos anteriores. Basicamente, o principal objetivo dessas ferramentas é realizar operações escaláveis para *clusters*, sendo uma boa solução independentemente da quantidade de nodos que formam o *cluster*. O System Imager [System-Imager 2004] é uma ferramenta desse grupo. Ele oferece um conjunto de *shell scripts* para a manutenção de uma mesma imagem de sistema operacional nos nodos do *cluster*. Os *scripts* utilizam-se de algoritmos de transferência, como, por exemplo, o proporcionado pelo aplicativo *rsync* [Group 2004a]. Essa funcionalidade é essencial durante manutenções no *cluster* quando apenas um pequeno conjunto de arquivos foi alterado e uma completa reinstalação do nodo mostra-se desnecessária.

Da perspectiva de gerenciamento de redes, as ferramentas citadas anteriormente podem não ser sempre identificadas como ferramentas de gerenciamento no sentido estrito da palavra, mas em termos de gerenciamento de *clusters* não é raro encontrar esta denominação. Um problema mais crítico, entretanto, é o fato de tais ferramentas não poderem ser integradas em uma solução baseada em SNMP sem complexas adaptações. Das ferramentas citadas, apenas SIMONE tem facilidades para integração com SNMP, porém o foco dessa ferramenta é o gerenciamento de sistemas distribuídos em geral, não contendo informações específicas para *clusters* como, por exemplo, alocação dos nodos, nem lidando com problemas como a interação entre a rede interna e externa ao *cluster*.

3. MIBs para Gerenciamento de *Clusters* e Respectivo Agente SNMP

O sistema para gerenciamento de *clusters* desenvolvido é baseado na existência de uma aplicação de gerenciamento que controla agentes SNMP internos aos *clusters* de interesse. As informações de gerenciamento de *cluster* que cada agente SNMP suporta são definidas em MIBs para *clusters*.

As MIBs definidas para gerenciamento de *clusters* são resultado da análise das informações de gerenciamento necessárias para o gerenciamento de *clusters*, mas também da análise de outras MIBs previamente existentes. As MIBs MIB-II [Mccloghrie and Rose 1991] e HostResources [Grillo and Waldbusser 1993], principalmente, permitem que diversos dados de gerenciamento de *clusters* não precisem ser definidos novamente em outras MIBs. Da MIB-II foi aproveitado o grupo *system*, que disponibiliza informações gerais da máquina. As informações de rede são obtidas no grupo *interfaces*, que apresenta dados sobre as interfaces de rede da máquina. Da MIB HostResources foram aproveitadas informações de três grupos:

- *hrSystem*: provê informações como tempo que a máquina está em operação (*hrSystemUptime*) e a data do sistema (*hrSystemDate*).
- *hrStorage*: provê na tabela *hrStorageTable* informações como tamanho total e uso de memória RAM, de memória *swap*, e de disco.
- *hrDevice*: provê na tabela *hrPartitionTable* dados sobre partições de disco.

Complementarmente, as funcionalidades mais específicas dos *clusters*, não cobertas pela MIB-II e HostResources, foram definidas em duas novas MIBs: uma MIB voltada ao gerenciamento de nodos e outra MIB voltada ao gerenciamento do *front-end*. As funcionalidades suportadas nessas MIBs, na atual versão, incluem características de dois grupos de gerenciamento descritos na seção anterior: ferramentas de monitoramento e ferramentas de alocação de nodos para usuários (a investigação de funcionalidades para automação de tarefas em *clusters* via SNMP está em andamento). Tais funcionalidades são úteis por proporcionarem informações gerais sobre o *cluster* e seu funcionamento, possibilitando a identificação de sobrecarga de recursos (CPU, memória e disco), o suporte à alocação de nodos e o controle de temperatura desses.

3.1. As MIBs ClusterNode e ClusterFrontEnd

A Figura 1 apresenta as MIBs ClusterNode e ClusterFrontEnd para gerenciamento dos nodos e do *front-end* de um *cluster*.

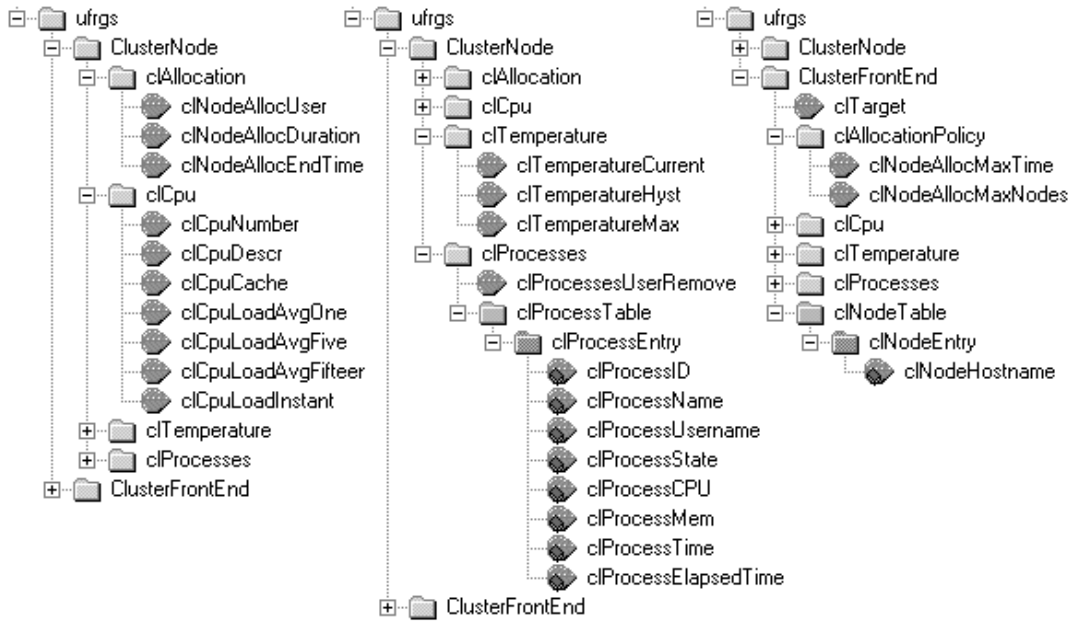

Figura 1. MIBs ClusterNode e ClusterFrontEnd

Como pode ser observado, a MIB ClusterNode é formada por quatro grupos (*clAllocation*, *clCpu*, *clTemperature* e *clProcesses*).

A alocação e liberação de nodos são realizadas através do grupo *clAllocation*. O objeto de leitura e escrita *clNodeAllocUser* contém ou o nome do usuário para o qual o nodo está reservado, ou o valor "ALL" caso o acesso esteja liberado. Depois da alocação, nenhum usuário, exceto aquele que reservou o nodo, está apto a efetuar *login*. Para liberar o nodo, isto é, permitir que todos os usuários voltem a ter acesso ao nodo, o administrador deve atribuir o valor "ALL" novamente em *clNodeAllocUser*. O objeto *clNodeAllocDuration* deve ser ajustado para indicar o número de horas que a reserva irá durar. Tal ajuste deve ser realizado antes da efetiva reserva do nodo através de *clNodeAllocUser*. Por fim, o objeto de leitura *clNodeAllocEndTime* informa data e hora em que a reserva se encerrará, de acordo com o valor atribuído a *clNodeAllocDuration*.

O grupo *clCpu* expõe dados a respeito da(s) CPU(s) do nodo. *clCpuNumber* informa o número de CPUs do nodo e *clCpuDescr* apresenta uma descrição da(s) CPU(s). A quantidade de memória *cache* de cada processador é apresentada em *clCpuCache*. Os dados de utilização de CPU são fornecidos por quatro objetos: *clCpuLoadAvgOne*, *clCpuLoadAvgFive*, *clCpuLoadAvgFifteen* e c*lCpuLoadInstant*. Os três primeiros apresentam a carga média de processamento no nodo no último minuto, nos últimos cinco minutos e nos últimos quinze minutos, respectivamente. Já o objeto *clCpuLoadInstant* informa a carga de processamento em um dado instante. Esses indicativos são calculados através de uma média aritmética entre as porcentagens de uso de cada CPUs do nodo. Por exemplo, um valor de 0,5 indica que 50% da capacidade total das CPUs do nodo está em uso.

O grupo *clTemperature* informa os valores obtidos a partir dos sensores de temperatura da placa mãe do nodo, o qual deve possuir o *software lmsensors* [Group 2004b] devidamente configurado. O objeto de leitura *clTemperatureCurrent* informa a temperatura atual do processador em graus Celsius. O objeto de leitura *clTemperatureHyst* indica um limiar de alerta a partir do qual cuidados devem ser tomados. *clTemperatureMax* informa o limite máximo permitido para temperatura, o qual, quando ultrapassado, requer o desligamento da máquina. Como os limiares de temperatura só podem ser alterados por complexas intervenções manuais em cada nodo, os objetos *clTemperatureHyst* e *clTemperatureMax* só podem ser utilizados para se obter tais valores, mas não são utilizados para sua alteração. Para o gerenciamento adequado, o software gerente deve constantemente monitorar o valor de *clTemperature* e compará-lo com *clTemperatureHyst* e *clTemperaturaMax* de forma a identificar quando a temperatura corrente ultrapassa os limiares críticos. Opcionalmente, uma *trap* SNMP poderia ser gerada para indicar ao software gerente que a temperatura do nodo ultrapassou os limiares críticos. Entretanto, como *traps* SNMP são mensagens sem confirmação e podem ser perdidas, optou-se, no atual estágio do trabalho, por assumir que os nodos serão monitorados por um processo de *pooling* a partir do software de gerenciamento. O suporte a *traps* no agente deverá ser investigado no futuro.

As informações relativas aos processos de um nodo estão no grupo *clProcesses*, o qual é formado por c*lProcessesUserRemove* e por *clProcessTable*. A folha *clProcessesUserRemove* é utilizada para encerrar todos os processos de um usuário que estão rodando no nodo (através da escrita da string do nome do usuário em questão), ou então para encerrar todos os processos de todos os usuários (através da escrita da string

"ALL"). Os processos também podem ser encerrados individualmente através do objeto *clProcessState* (explicado a seguir). A tabela *clProcessTable* apresenta informações sobre os processos rodando no nodo. Parte dos dados fornecidos por essa tabela é implementada na MIB HostResources, porém outros dados importantes (e.g. usuário dono do processo e o tempo que o processo está rodando) não são disponibilizados, o que evidencia a necessidade de extensão das informações da HostResources através da tabela *clProcessTable* na ClusterNode MIB. Uma otimização realizada baseia-se no fato de que, no contexto de *clusters*, muitas vezes o interesse recai apenas sobre os processos de usuário que estão rodando, sem haver interesse especial nos processos do sistema operacional. Assim, optou-se por implementar inclusive os dados já suportados MIB HostResources, entretanto exportando apenas os processos de usuário. Isso economiza tráfego na rede, pois apenas uma parte da lista de processos precisa ser exportada.

Cada linha da tabela *clProcessTable* referencia um processo em execução no nodo. *clProcessID* indexa cada processo na tabela, além de informar o identificador do processo (PID). O objeto *clProcessName* indica o nome do processo e o objeto *clProcessUsername* indica o *username* do dono do processo. O estado do processo (e.g. ativo, em entrada/saída, parado ou *zoombie*) é informado pelo objeto de leitura e escrita *clProcessState*. Esse objeto também é usado para possibilitar o encerramento de processos quando o valor "*kill*" for atribuído ao objeto (esta abordagem foi baseada na MIB HostResources). Adicionalmente, *clProcessCPU* e *clProcessMem* informam percentualmente quanto de CPU e de memória o referido processo está ocupando. O objeto *clProcessTime* informa o tempo que o processo já utilizou o processador e o objeto *clProcessElapsedTime* informa há quanto tempo este foi iniciado.

A MIB ClusterFrontEnd assemelha-se muito com a MIB ClusterNode, sendo formada por um objeto escalar (*clTarget*), por quatro grupos (*clAllocationPolicy*, *clCpu*, *clTemperature, clProcesses*) e por uma tabela (*clNodeTable*). As diferenças entre as duas MIBs estão no objeto *clTarget*, no grupo *clAllocationPolicy* e na tabela *clNodeTable*. O objeto *clTarget* desempenha uma função importante na lógica de gerenciamento do cluster: ele está relacionado ao processo de redirecionamento das solicitações SNMP aos nodos do *cluster* (funcionalidade descrita na subseção 3.2).

O grupo *clAllocation* da MIB ClusterNode foi substituído pelo grupo *clAllocationPolicy* na MIB ClusterFrontEnd. Isso porque, ao contrário do que acontece nos nodos, não faz sentido o *front-end* ser reservado, pois esse é o local onde qualquer usuário deve poder se conectar para programar, compilar e testar suas aplicações. Por outro lado, cabe ao *front-end* armazenar a política de alocação do *cluster*. Uma política de alocação é um conjunto de regras que visa um melhor compartilhamento dos recursos, fazendo com que o máximo de usuários consigam utilizar o *cluster* ao mesmo tempo, enquanto se tenta reduzir o tempo de espera para obter acesso aos recursos. Assim, o grupo *clAllocationPolicy* armazena a política de alocação, e contém as folhas de leitura e escrita *clNodeAllocMaxTime* e *clNodeAllocMaxNodes*, as quais indicam, respectivamente, o máximo de horas e de nodos permitidos em uma reserva.

Por fim, a tabela *clNodeTable*, inexistente na MIB ClusterNode, armazena no objeto *clNodeHostname* o valor de *sysName* da MIB-II dos nodos. Tanto *clTarget* quanto *clNodeHostname* são objetos chave para o encaminhamento de mensagens SNMP aos nodos de interesse. O uso conjugado desses objetos é apresentado à frente.

3.2. Implementação dos Agentes SNMP

Os agentes SNMP foram construídos para ambiente Linux, implementados em linguagem C e utilizando as facilidades para construção de agentes do pacote NET-SNMP [NET-SNMP 2004]. Para evitar a captura das *strings* de comunidade SNMP utilizadas pelo administrador no gerenciamento de *clusters*, os agentes são configurados de forma a utilizarem o protocolo SNMPv3 [Blumenthal e Wijnen 1998], o qual dá suporte a uma série de itens de segurança, tais como autenticação de usuário e garantia de integridade dos dados.

Dois agentes SNMP diferentes, mas complementares, foram implementados: um agente executado internamente em cada nodo do *cluster* e um agente executado no *front-end*. Logo, os elementos de um *cluster* (nodos ou *front-end*) possuem agentes SNMP. Entretanto, apenas o agente do *front-end* é acessado diretamente pelo sistema de gerenciamento. Os agentes nos nodos só podem ser acessados através do agente do *front-end*. Diz-se então que o agente do *front-end*, além de fornecer suporte para o gerenciamento próprio *front-end*, é um também um *proxy* SNMP.

A necessidade de implementação de um *proxy* SNMP vem de uma dificuldade inerente à configuração da maioria dos *clusters*: geralmente, a única máquina acessível a partir da rede externa na qual o *cluster* está inserido é o *front-end*, estando os nodos do *cluster* isolados em uma rede interna, "escondidos" atrás do *front-end*. Assim, a única forma de acessar os nodos é através do acesso ao *front-end*. Cabe ressaltar que tal configuração dos *clusters* é adotada por diversas razões. Entre elas pode-se citar a concentração da autenticação e dos *softwares* do *cluster* no *front-end*, um melhor controle de segurança através de um ponto único de acesso, a reserva da rede interna do *cluster* apenas para este e, por fim, a utilização de endereços IP falsos para evitar o uso de IPs reais para cada nodo. Assim, o objetivo do *proxy* SNMP é permitir que o *front-end* realize a interação entre a rede externa e interna ao *cluster*. Requisições SNMP para acesso aos nodos devem ser feitas para o *front-end* e este, identificando o real destino da requisição, deve tratar o pacote e repassá-lo para o nodo na rede interna do *cluster*.

A Figura 2 apresenta um exemplo de interação através do *proxy* SNMP. Nesse exemplo, um software gerente está interessado em ler o valor do objeto *clCpuNumber* da MIB ClusterNode. Antes, entretanto, o gerente precisa manipular o objeto *clTarget* da MIB ClusterFrontEnd de forma que o agente SNMP do *front-end* comece a repassar mensagens SNMP, na rede interna do *cluster*, ao nodo 2 de interesse.

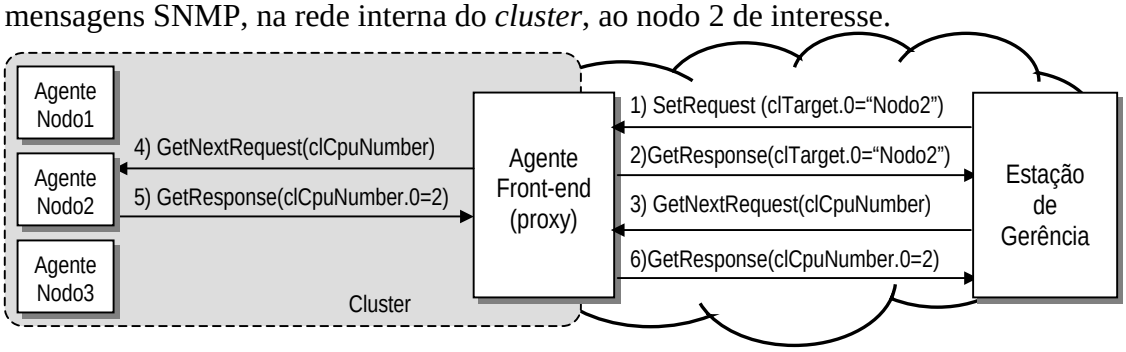

Figura 2. Funcionamento básico do proxy SNMP

Ao receber uma requisição SNMP, o agente do *front-end* deve, antes de proceder com o processamento SNMP padrão, consultar o valor do objeto *clTarget* interno. Se o valor

de *clTarget* for igual a um dos valores armazenados por *clNodeHostname* da tabela *clNodeTable*, então o agente do *front-end* repassa a mensagem SNMP para o nodo identificado em *clTarget*, e aguarda por uma resposta que será então enviada de volta ao *software* gerente que fez a solicitação original. Caso o valor de *clTarget* não corresponda a nenhuma entrada *clNodeHostname*, então a mensagem SNMP é tratada localmente no *front-end* e a respectiva resposta enviada de volta ao gerente. O *front-end*, ao encaminhar uma mensagem para um nodo, descobre o endereço IP desse nodo através do processo de resolução de nomes da rede interna do *cluster*, que no caso dos *clusters* ocorre, normalmente, através de um mapeamento estático baseado no arquivo */etc/hosts* existente no sistema operacional do *front-end.*

Esse funcionamento indica que um *software* gerente necessita, antes de interagir com um nodo do *cluster*, alterar o valor de *clTarget* no *front-end* para informar o nodo que efetivamente tratará as requisições SNMP. Na Figura 2, (1) o gerente do *cluster* solicita a escrita do valor "Nodo 2" em *clTarget* e (2) recebe uma resposta confirmando a operação. Logo a seguir, todas as mensagens SNMP que chegarem ao *front-end* serão encaminhadas ao nodo 2 do *cluster*. No exemplo, (3) o gerente solicita a leitura de *clCpuNumber*. Essa solicitação é repassada (4) ao agente no nodo 2, que responde (5) informando o número de CPUs do nodo. Por fim, a resposta (6) devolvida ao gerente.

3.3. Suporte às MIBs ClusterNode e ClusterFrontEnd

Nesta subseção é descrita a forma como os objetos das MIBs ClusterNode e ClusterFrontEnd são implementados junto ao *cluster*.

O objeto *clNodeAllocUser* do grupo *clAllocation* manipula o arquivo */etc/security/access.conf*. Esse arquivo faz parte de um mecanismo de controle de acesso (*login*) em sistemas Linux baseados em regras de acesso. Caso não haja regra alguma que proíba o acesso à máquina, todos os usuários estarão aptos a acessá-la. Quando um nome de usuário é escrito no objeto *clNodeAllocUser*, o arquivo de configuração */etc/security/access.conf* é alterado para negar o acesso a qualquer usuário que não seja o beneficiado pela reserva. Quando o objeto *clNodeAllocUser* recebe a *string* "ALL" através de uma escrita, o arquivo *access.conf* é esvaziado. A linha a seguir é inserida no arquivo de configuração para permitir o acesso apenas do administrador (*root*) e do usuário em questão (*username*).

```
-:ALL EXCEPT root username
```

Além disso, o valor do objeto *clNodeAllocDuration* é consultado para que um agendamento da execução de um *script* seja executado. Após agendar o procedimento de liberação do nodo, o objeto *clNodeAllocEndTime* é alterado para conter a data e a hora em que a reserva do nodo acabará. Quando o *script* de liberação for executado após *clNodeAllocDuration* horas, o conteúdo de *clNodeAllocDuration* e *clNodeAllocEndTime* é apagado.

Os dois objetos de escrita *clNodeAllocMaxTime* e *clNodeAllocMaxNodes*, que formam o grupo *clAllocationPolicy* situado no *front-end*, têm seus valores armazenados, respectivamente, nos arquivos */etc/clallocmaxtime.conf* e */etc/clallocmaxnodes.conf*. O arquivo de sistema */proc/cpuinfo* provê as informações relacionadas ao grupo *clCpu* contidas nos objetos *clCpuNumber*, *clCpuDescr*, *clCpuLoadAvgOne*, *clCpuLoadAvgFive* e *clCpuLoadAvgFifteen*. Os dois primeiros são obtidos através do arquivo */proc/cpuinfo* e os três últimos através do arquivo */proc/loadavg*. O valor do

objeto *clCpuLoadInstant* é obtido através da soma dos percentuais de uso de CPU relativos a cada processo rodando, obtidos através do comando *ps*.

As informações que compõem o grupo *clTemperature* são obtidas a partir do diretório */proc/sys/dev/sensors*, lendo o arquivo *temp* localizado dentro do subdiretório correspondente ao sensor contido na placa-mãe. Tal subdiretório deve ser indicado manualmente pelo administrador através do arquivo de configuração */etc/clsensors.conf*. A instalação e configuração do pacote *lmsensors* é de total responsabilidade do administrador do *cluster*.

Para encerrar processos quando o objeto *clProcessUserRemove* for alterado, é utilizada uma combinação dos comandos *ps*, *grep* e *kill* do Unix. A tabela *clProcessTable* é preenchida de acordo com a saída do comando "*ps axo pid,user,%cpu,time*". Cada linha da saída desse comando fornece uma linha da tabela, e cada coluna da saída fornece um objeto da tabela *clProcessTable*. Novamente, o encerramento de processos na tabela é feito através do comando *kill*. A tabela *clNodeTable* reflete o conteúdo de um arquivo de configuração situado no *front-end* em */etc/clnodes.conf* e deve ser preenchido manualmente pelo administrador do *cluster*.

4. Estudo de Caso: O Sistema de Gerenciamento SNMP para *Clusters*

O sistema para gerenciamento de *clusters* via SNMP é composto por um conjunto de scripts PHP [PHP 2004] disponibilizados aos administradores através de uma interface Web. O uso de gerenciamento baseado na Web é amplamente discutido e suas vantagens são bem conhecidas e divulgadas na literatura [Martin-Flatin 2001]. O uso de PHP como linguagem base da implementação se justifica pelo fato de a linguagem ser simples, o que possibilita um rápido desenvolvimento e uma manutenção facilitada. Mais importante que isso, entretanto, é o fato de o PHP já possuir suporte ao SNMP através de uma API de fácil utilização.

A interface gráfica permite uma interação progressiva do administrador com o sistema. Inicialmente, as informações visualizadas permitem a verificação de dados globais do *cluster* (Figura 3), como o *hostname* do *front-end*, o número de nodos e de processadores (obtidos com o grupo *clNodeTable* da MIB ClusterFrontEnd e com o grupo *clCpu* da MIB ClusterNode), além da quantidade total de memória e de disco (obtidos junta à MIB HostResources).

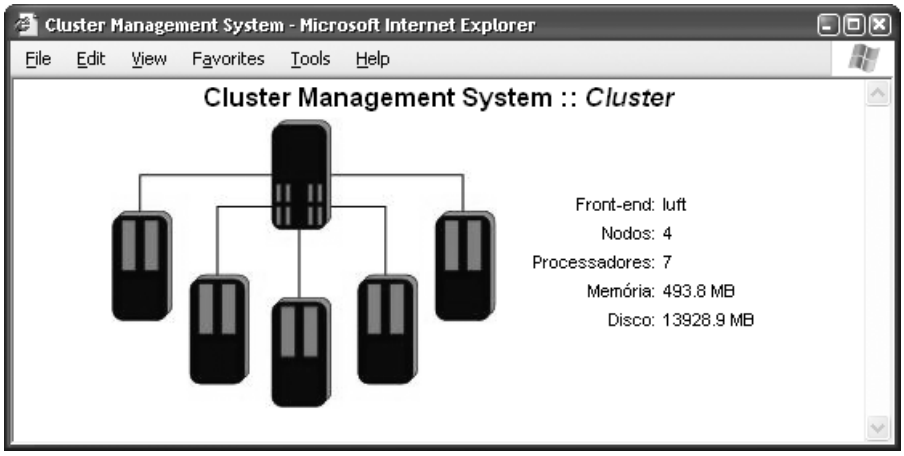

Figura 3. Tela inicial do Sistema de Gerenciamento

Acessando a imagem que ilustra o *cluster*, o administrador acessa uma página onde os nodos são apresentados (Figura 4). Para cada nodo é mostrada uma imagem descritiva, através da qual pode-se obter duas informações: carga de trabalho e estado de reserva. De acordo com a carga de CPU, a imagem do nodo assume uma cor diferente (verde, amarelo ou vermelho), para representar carga baixa (abaixo de 30%), média (entre 30% e 80%) e alta (acima de 80%), respectivamente[2]. Outra informação fornecida é o estado de reserva do nodo, que, se estiver reservado, mostrará a imagem de um cadeado. Esses dados são obtidos nos grupos *clCpu* e *clAllocation* da MIB ClusterNode.

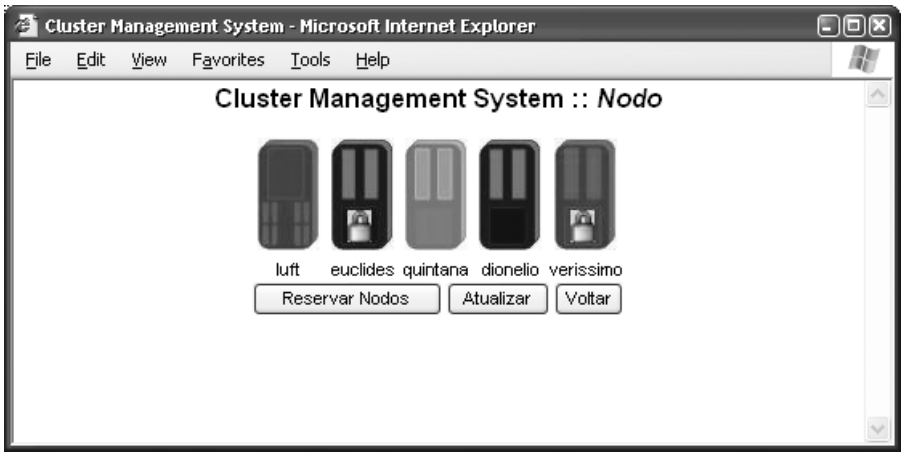

Figura 4. Tela que apresenta os nodos do *cluster*

O botão "Reservar Nodos" apresenta uma interface (Figura 5) que, além de reservar nodos, informa se um nodo está reservado, qual foi o usuário que efetuou a reserva e a data e hora em que a reserva se encerrará. Com essas informações, obtidas a partir da MIB ClusterNode, pode-se efetuar uma alocação, escolhendo a duração da reserva e os nodos envolvidos. As opções de duração de reserva são oferecidas de acordo com a política de alocação do *cluster* e o limite de nodos é informado abaixo da tabela de nodos (essas informações são recuperadas junto ao grupo *clAllocationPolicy* da MIB ClusterFrontEnd). Como as requisições somente serão aceitas enquanto a política não for transgredida, se uma reserva ultrapassar tal limite ela será atendida parcialmente.

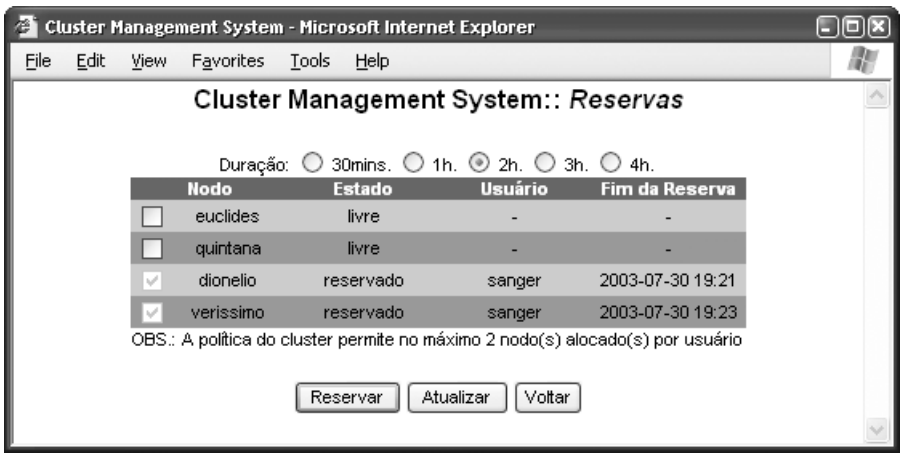

Figura 5. Tela para reserva de múltiplos nodos

[2] Os limiares de utilização dos nodos foram definidos empiricamente.

É importante ressaltar que as interfaces para o administrador do *cluster* e para os usuários diferem, pois o primeiro deve poder alterar a política de reservas do *cluster*. Assim, apenas para usuários com privilégio de administrador é apresentado o botão "Alterar Política". Pressionando-o, é mostrada uma nova interface onde se pode ajustar a política de alocação do *cluster*.

Voltando à Figura 4, pode-se obter informações específicas sobre cada nodo ou sobre o *front-end*, clicando-se em suas imagens. Abre-se então uma página que concentra as seis principais opções de conjuntos de dados disponibilizados. São elas: informações de *hardware*, de processos em execução, de temperatura, de utilização de recursos, de reserva do nodo, e das partições do(s) disco(s) do nodo. Para acessar cada categoria de informações, basta escolher a opção desejada (Figura 6).

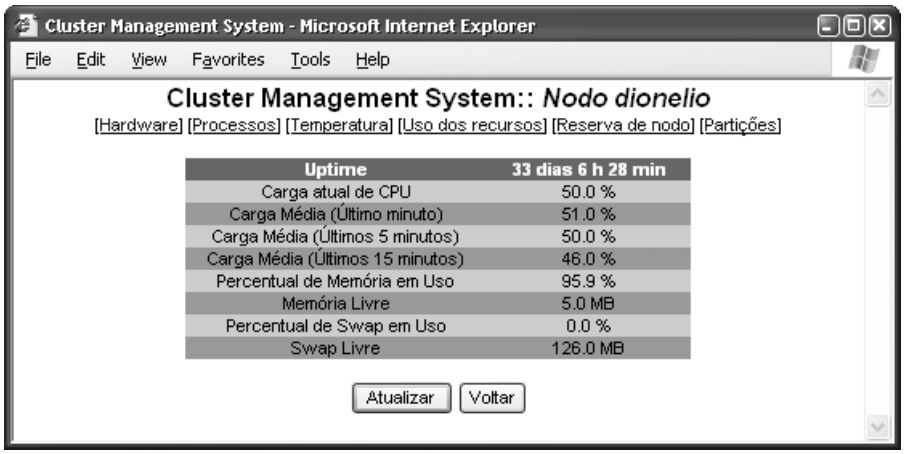

Figura 6. Interface que apresenta informações sobre uso de recursos do *cluster*

Como dito anteriormente, a primeira opção é denominada "*Hardware*". Nessa opção o usuário visualiza um detalhamento do *hardware* da máquina verificando informações a respeito da CPU do nodo (quantidade, descrição e *cache*), informações sobre o total de memória, de disco e de *swap*, e uma lista das interfaces de rede disponíveis. Através dessa interface, o usuário tem condições de caracterizar por completo o nodo, sabendo assim qual o ferramental disponível onde sua aplicação irá ser executada. Os dados dessa opção são obtidos via acessos às MIBs HostResources e ClusterNode do nodo.

A opção "Processos" apresenta uma lista com os processos de usuário em execução na máquina. O nome do processo, o dono (usuário), o identificador (PID), o estado, o uso percentual de CPU e de memória, o tempo desde o disparo e o tempo de execução são apresentados. É possível também finalizar processos, selecionando um *checkbox* correspondente e confirmando o encerramento no botão *Kill*. Os dados apresentados por essa opção estão ligados ao grupo *clProcesses* da MIB ClusterNode.

A opção "Temperatura" apresenta as informações da temperatura atual do nodo, o limiar de temperatura a partir do qual este requer atenção e a temperatura máxima suportada pelo nodo, obtidas através do grupo *clTemperature* da MIB ClusterNode. Para tanto o nodo deve possuir *hardware* com suporte a sensores de temperatura e esses devem estar devidamente configurados.

A quarta opção é "Uso dos recursos". Nessa opção, o usuário obtém informações, obtidas das MIBs HostResources e ClusterNode, sobre o estado atual de utilização de recursos do nodo, como CPU, memória, disco e memória *swap*. Em

relação ao uso de CPU são mostradas quatro porcentagens de carga: a primeira indica o percentual de uso no momento; a segunda, a terceira e a quarta apresentam a carga média (*loadavg*) de utilização da CPU nos últimos um, cinco e quinze minutos, respectivamente. Sobre a utilização de memória convencional e de memória *swap* são apresentados o percentual em uso e a informação de quantos *megabytes* estão disponíveis no instante da consulta.

A opção "Reserva de nodo" trata a reserva de nodos separadamente, um a um, ao contrário da apresentada na Figura 5. Caso o nodo já esteja reservado, o usuário é informado de quando a reserva terminará. Finalmente, na sexta opção, denominada "Partições", são apresentadas as informações a respeito das partições de disco ativas na máquina. Para cada partição é mostrado o seu tamanho total, a quantidade de espaço em uso e o percentual de uso, obtidos através de acessos à MIB HostResources.

5. Conclusões e Trabalhos Futuros

O conjunto de dados de gerenciamento definidos nas MIBs ClusterNode e ClusterFrontEnd abrange boa parte das necessidades de um gerente para *clusters*. Outras necessidades são cobertas, de forma complementar, por demais MIBs como a MIB-II e a HostResources. Algumas informações de gerenciamento acabam sendo encontradas em mais de uma MIB, como por exemplo, os dados relativos aos processos, que são encontrados na MIB HostResources e na MIB ClusterNode. Entretanto, o tráfego na rede interna é menor quando os processos são recuperados através da ClusterNode porque só os processos de usuário são retornados, o que não aconteceria se fosse usada a HostResources, que retorna também os processos do sistema. Além disso, o suporte para finalização de processos na ClusterNode é mais apropriado ao gerenciamento de *clusters*, pois é possível se encerrar processos individualmente, por usuário, ou então encerrar todos os processos de um usuário em um nodo.

Em relação à arquitetura, o uso de um *proxy* SNMP impõe ao *front-end* uma sobrecarga de CPU relativa ao processamento, à definição de destino e à remontagem do pacote a ser redirecionado. Porém, como as funcionalidades do *front-end* são, em geral, meramente administrativas, a sobrecarga é bem tolerável. Além disso, a abordagem de se ter o agente SNMP do *front-end* desempenhando funções de *proxy* SNMP permite a instalação de suporte a outras MIBs SNMP nos nodos, sem a necessidade de alteração do agente no *front-end.*

O sistema desenvolvido foi aplicado ao gerenciamento de dois clusters e encontra-se atualmente em uso nos mesmos. Considerando que ambos os clusters são baseados em Linux, o processo de instalação da infra-estrutura SNMP envolve a instalação do NET-SNMP e do suporte desenvolvido para as MIBs ClusterNode e ClusterFrontEnd em cada nodo e no *front-end* do *cluster*. Já a instalação do sistema de gerenciamento é mais simples, pois ele é constituído apenas por um conjunto de scripts PHP. Logo, necessita-se apenas de uma máquina com Apache e PHP (com suporte SNMP habilitado) instalados, e respectiva cópia dos scripts que formam o sistema.

A conclusão mais importante, na nossa opinião, é que a solução desenvolvida permite observar que o gerenciamento de *clusters* de alto desempenho pode ser baseado em um protocolo de gerenciamento de redes, no caso o SNMP. A infra-estrutura de agentes criada permite então que o gerenciamento de *clusters* seja facilmente integrado ao gerenciamento do restante da rede, já que as plataformas de gerenciamento de redes

utilizam SNMP para acesso aos dispositivos: agora elas também podem utilizar SNMP para acesso aos *clusters.* O uso do sistema desenvolvido se mostrou apropriado para a realização das tarefas administrativas mais freqüentes, como a alocação/liberação de nodos, acompanhamento de processos, e verificação do estado do cluster para determinação de uma estratégia de distribuição de execução de tarefas adequada. Uma tarefa também freqüente é a execução agendada de scripts para manutenção do *cluster*. Essa tarefa, entretanto, ainda não é suportada no sistema desenvolvido. O uso de SNMP para implementação de tal suporte poderia ser alcançado, por exemplo, através do uso da MIB Script [Levi and Schoenwaelder 1999]. Logo, o suporte a *scripts* através da MIB Script precisa ser verificado e, principalmente, seu impacto nas infra-estruturas internas do *cluster* observado. A monitoração contínua de *clusters* através do sistema desenvolvido é também um trabalho em andamento. O objetivo, nesse caso, é verificar, através de gerenciamento baseado em *pooling*, o histórico de uso dos recursos.

Referências

Blumenthal, U. e Wijnen, B. (1998) "User-based Security Model (USM) for version 3 of the Simple Network Management Protocol (SNMPv3)", IETF, RFC (2264).

Buyya, R., High performance cluster computing: architectures and systems, Upper Saddle River: Prentice Hall PTR, 1999.

Case, J., Fedor, M., Schoffstall, M. e Davin, J. (1990) "A Simple Network Management Protocol (SNMP)", IETF, RFC (1157).

Grillo, P. e Waldbusser, S. (1993) "Host Resources MIB", IETF, RFC (1514).

Group, T. R. (2004a) "Rsync", http://samba.anu.edu.au/rsync, Julho.

Group, T. L. (2004b) "Hardware Monitoring by lmsensors", http://secure.netroedge.com/ lm78/docs.html, Julho.

Levi, D. e Schoenwaelder, J. (1999) "Definitions of Managed Objects for the Delegation of Management Scripts", IETF, RFC (2592), Maio.

Martin-Flatin, J.P. e Anerousis, N. (2001) "Web-Based Management" (guest editorial), In: Journal of Network and System Management (JNSM) - Special Issue on Web-Based Management, Vol. 9, No. 1, Março.

Massie, M. (2003) "Ganglia - distributed monitoring and execution system", http://ganglia.sourceforge.net, Maio.

Mccloghrie, K. e Rose, M. (1991) "Management Information Base for Network Management of TCP/IP-based internets: MIB-II", IETF, RFC (1213).

NET-SNMP (2004). "The NET-SNMP Project", http://net-snmp.sourceforge.net, Julho.

PBS (2004) "PBS - Portable Batch System", http://www.openpbs.org, Maio.

PHP (2004) "PHP", http://www.php.net, Julho.

Subramanyan, R., Miguel-Alonso, J. e Fortes, J. A. B. (2000) "Design and evaluation of a SNMP-based monitoring system for heterogeneous, distributed computing", School of Electrical and Computer Eng, Purdue University, TR-ECE. (00-11).

System-Imager (2004) "System Imager", http://systemimager.org, Maio.