Sistema de Gerência para um Ambiente de Processamento Paralelo e Distribuído de Alta Velocidade

André Mello Barotto
andre@npd.ufsc.br

Sistema de Gerência para um Ambiente de Processamento Paralelo e Distribuído de Alta Velocidade

André Mello Barotto
andre@npd.ufsc.br

Adriano de Souza
adriano@npd.ufsc.br

Carlos Becker Westphall
westphal@lrg.ufsc.br

UFSC - CTC - INE – LRG
Laboratório de Redes e Gerência
Cx. Postal 476 CEP 88040-970 Florianópolis - SC – Brasil
Phone: +55 48 231-9739 FAX: +55 48 231-9770

Resumo

Java, WWW e CORBA são tecnologias que tem sido empregadas com grande sucesso na área de gerência de redes. No entanto, tais tecnologias não sobrepõem padrões de gerenciamento existentes como o CMIP e SNMP. Dentro deste contexto, este artigo descreve o uso destas tecnologias na implementação de um sistema de gerência distribuída para um conjunto de estações (*cluster*) destinadas ao processamento paralelo e distribuído. Este sistema de gerência é composto por: um grupo de objetos gerenciados, um agente de contabilização, uma aplicação de gerência Web e um agente SNMP.

Abstract

Java, WWW and CORBA are technologies that have been used with great success in the field of network management. However, these technologies do not overlap with already existing ones, such as CMIP and SNMP. In this context, this paper makes use of these technologies to implement a distributed system for the management of a cluster of stations that is destined to distributed and parallel processes. This management system is composed by: a group of managed objects, an accounting agent, a Web managing application and an SNMP agent.

Palavras Chave: CORBA, SNMP, Java, SNMP, Gerência de Redes.

1 INTRODUÇÃO

A linguagem Java, o WWW (*World Wide Web*) e a CORBA (*Common Object Request Broker Architecture*) são tecnologias complementares que, quando utilizadas em conjunto, oferecem um poderoso mecanismo para o desenvolvimento de aplicações distribuídas [10]. Tais tecnologias podem ser utilizadas com grande sucesso para a construção de aplicações distribuídas de gerenciamento, sem no entanto, sobrepor padrões de gerenciamento existentes como o SNMP

(*Simple Network Management Protocol)* ou o CMIP (*Common Management Information Protocol)*[10][11][16][2].

A tecnologia Web, quando aplicada na área de gerência de redes, permite que um administrador possa monitorar e configurar equipamentos conectados em sua rede, a partir de praticamente qualquer computador pertencente à Internet [1][8][9][11][12]. Cada vez mais, surgem equipamentos e *softwares* capazes de serem monitorados e configurados a partir de *browsers Web*, como: *switches*, roteadores, servidores Web, entre outros [3][7]. Estas características tornam o *browser* uma preciosa ferramenta para auxiliar o gerenciamento de redes através da Internet e Intranets.

Uma arquitetura que pode ser utilizada com muito sucesso em conjunto com a tecnologia Web é a CORBA [10][13]. Esta arquitetura permite que uma aplicação cliente possa instanciar objetos em uma máquina servidora, independente do hardware que esta máquina possua ou da linguagem de programação utilizada na implementação dos objetos [13][14][15]. Em virtude deste fato, ela pode ser empregada com grande sucesso na área de gerência de redes para a instanciação de objetos gerenciados [5][4][10][16].

Dentro deste contexto, o objetivo primordial deste artigo é descrever um conjunto de experimentos e implementações, visando a aplicação prática da arquitetura CORBA, do WWW, da linguagem Java e do SNMP direcionados para a área de gerência de redes. Visando o cumprimento deste objetivo, definiu-se como meta inicial a implementação de um sistema de gerência para um *cluster* de estações na Universidade Federal de Santa Catarina (UFSC). No escopo deste trabalho, um *cluster* de estações consiste em um conjunto de estações de trabalho interligadas por uma rede de alta velocidade e que possuem uma base comum de usuários. O *cluster* implantado na UFSC é composto por 20 estações IBM 43P, 2 SUN UltraSparc e 9 nós de um IBM SP2, interligados por uma estrutura de rede ATM 155Mbps e ETHERNET.

O *cluster* foi definido como o ambiente inicial para realização dos experimentos em virtude do fato de apresentar uma estrutura essencialmente distribuída. Neste caso, utilizando-se o *cluster* como ambiente a ser gerenciado, pode-se evidenciar de modo mais contundente os benefícios da utilização da CORBA, quando aplicada a área de gerência de redes.

Para efetuar gerência do *cluster*, foram implementados um conjunto de objetos que podem ser instanciados, através do protocolo IIOP (*Internet Inter-ORB Protocol*) [14][15], em qualquer máquina do *cluster*, afim de realizar as operações de gerência desejadas. Foram implementados também agentes para realizar a função de contabilização de recursos e para permitir a integração deste ambiente com aplicações de gerência SNMP. Para o gerenciamento Web deste ambiente, foi implementado um protótipo de uma aplicação de gerência, utilizando-se a linguagem Java.

Neste artigo, primeiramente é apresentada a estrutura lógica do sistema de gerência definido para o *cluster*. A seguir, são descritos de modo mais detalhado os objetos gerenciados, os agentes e a aplicação gerência que compõem o sistema. Finalmente, é realizada uma explanação sobre os resultados obtidos, dificuldades encontradas e os planos futuros para a continuidade deste projeto.

2 ESTRUTURA LÓGICA

Para o gerenciamento do *cluster*, deve-se definir primeiramente um mecanismo que permita a obtenção e modificação de parâmetros de gerência (Figura 1). Para tanto, foi definido um objeto denominado *Computer*. Este objeto é instanciado em cada máquina do *cluster*,

utilizando o protocolo IIOP/CORBA. Como o *cluster* caracteriza-se por ser um ambiente heterogêneo, em que as máquinas podem possuir diferentes *hardwares* e sistemas operacionais, um dos objetivos primordiais deste projeto é definir uma solução independente de plataforma para o ambiente. Em consequência deste fato, todas as aplicações e o ORB (*Object Request Broker*) foram implementados em Java. Neste sentido, foi escolhido como ORB a ser utilizado neste projeto o JORB. Este ORB integra o software *Bojangles* da IBM [4].

Com a finalidade de permitir o gerenciamento de parâmetros relacionados ao conjunto de estações do *cluster*, foi definido um objeto denominado *Cluster*. Este, por sua vez, instancia os objetos *Computer* em cada máquina do *cluster*, permitindo o gerenciamento de todas as máquinas que compõem este ambiente, através do envio de mensagens a um único objeto. Utilizando este mecanismo, um administrador pode obter e modificar informações em todas as máquinas do *cluster*, emitindo uma única mensagem ao objeto *Cluster*. É possível também, construir *applets* que, utilizando o protocolo IIOP, podem instanciar o objeto *Cluster* e obter informações como: quais as máquinas com menor ocupação de CPU dentro do *cluster*. Tais informações são muito importantes para um pesquisador decidir em quais máquinas executar os seus programas paralelos ou distribuídos.

Para a execução de tarefas dinâmicas, como o armazenamento de informações de *logs* ou o acompanhamento do *status* das máquinas, foi criado um agente, implementado em Java, que instancia o objeto *Cluster*. Este agente monitora constantemente o conteúdo de determinados atributos do objeto *Cluster* e, dependendo deste conteúdo, executa determinadas ações como, por exemplo, iniciar o armazenamento periódico da taxa de ociosidade de CPU das máquinas do *Cluster*, em uma base de dados relacional, permitindo um estudo detalhado da efetiva utilização destas máquinas.

As aplicações de gerência são implementadas como *applets* que utilizam o protocolo IIOP/CORBA para instanciar o objeto *Cluster* e emitir mensagens para este. O usuário pode acessar as aplicações de gerência, independente da localização do seu equipamento ou da sua plataforma, sendo necessário apenas, que este equipamento esteja conectado à rede e possua um *browser Web* com suporte ao Java instalado.

Afim de permitir a integração com outras aplicações de gerência, foi implementado um agente SNMP que age como um *gateway* entre os protocolos SNMP e IIOP. Este agente permite a integração deste ambiente com outras aplicações de gerência.

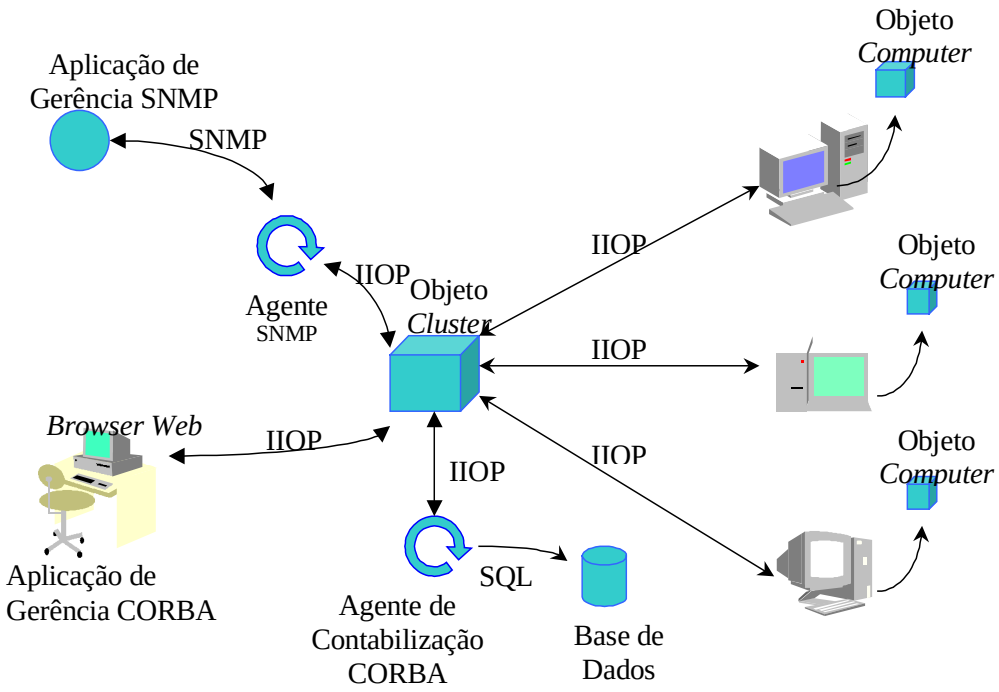

Figura 1 Estrutura Lógica do Sistema de Gerência

3 OBJETOS GERENCIADOS

Afim de permitir a gerência do *cluster*, foram definidos dois objetos que podem ser instanciados e consultados através do protocolo IIOP/CORBA. Estes objetos receberam a denominação de *Computer* e *Cluster*. O objeto *Computer* realiza a abstração de estações de trabalho individuais no *cluster*. O objeto *Cluster* realiza a abstração do *cluster* como um todo, representado todo o conjunto de estações deste ambiente. O objeto *Cluster* pode ser consultado pela aplicação de gerência, através de invocações de métodos, utilizando a arquitetura CORBA.

O objeto *Computer* é responsável por receber solicitações provenientes do objeto *Cluster*, realizar o processamento desejado e emitir as respostas requeridas.

O objeto *Cluster* tem a função de realizar a interface primária entre a aplicação gerente e os objetos gerenciados. O objeto *Cluster* instancia o objeto *Computer* em cada máquina do *cluster* e é responsável por receber mensagens provenientes da aplicação gerente, repassá-las para as máquinas desejadas, obter os resultados, processá-los e emitir as respostas encontradas.

3.1 OBJETO *COMPUTER*

O objeto *Computer* age como um componente intermediário entre o usuário e o sistema operacional da máquina. Este objeto tem a função de receber solicitações, emitir comandos para o sistema e retornar informações sobre a máquina em que está instanciado. O objeto *Computer* é implementado em Java, instanciado através do protocolo IIOP/CORBA e é capaz de executar,

praticamente, qualquer comando que um usuário poderia emitir através de um *shell Unix*. Utilizando-se deste recurso, é possível obter informações ou executar qualquer comando em uma máquina, através de uma simples invocação de método para o objeto *Computer*.

Como pode ser visto na *Figura 2*, a operação básica do objeto *Computer* resume-se a:

1. receber uma invocação de método, através do protocolo IIOP;
2. enviar uma requisição ao sistema na forma de um comando de *shell* UNIX, utilizando-se da classe *Runtime* da Java [13];
3. redirecionar os resultados obtidos para um arquivo temporário;
4. ler os resultados de um arquivo temporário e processá-los;
5. enviar os resultados encontrados para o cliente que requisitou as informações.

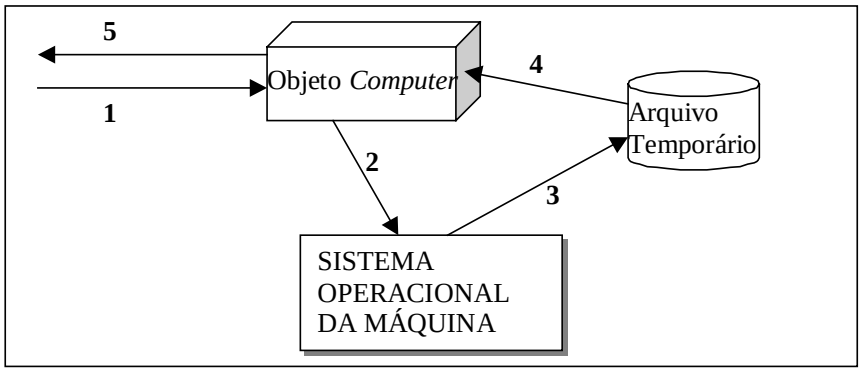

Figura 2 Operação Básica do Objeto *Computer*

3.1.1 IMPLEMENTAÇÃO DO OBJETO *COMPUTER*

Afim de implementar a estrutura básica para interação entre objetos instanciados através do protocolo IIOP e o sistema operacional de uma estação de trabalho, foi desenvolvido um protótipo do objeto *Computer*. Neste protótipo, o objeto *Computer* é capaz de obter a taxa de ociosidade da CPU de uma máquina, gerenciar um sistema de contabilização e retornar informações sobre o sistema operacional de tal máquina. A implementação deste protótipo foi realizada utilizando-se a *Java* e pode ser facilmente expandida para atender a novas funções de gerência, uma vez que a funcionalidade básica para a emissão de comandos ao sistema operacional é sempre a mesma.

Durante a construção do objeto *Computer*, foi definida primeiramente a interface deste objeto, utilizando-se a declaração *interface* da Java (Quadro 1). A partir desta definição, foram gerados *Stubs* e *Skeletons* [14], através da utilização dos utilitários *RunStubEmitter e RunSkeletonEmitter*, pertencentes ao *Bojangles* [4]. Foi realizada também a implementação dos objeto *Computer*. O código desta implementação foi distribuído entre as diversas máquinas que compõem o *Cluster*, afim de permitir a instanciação deste objeto nestas máquinas.

```
package agente;

import java.lang.Object;

public interface Computer {

   public boolean getStatusContab();
   public void startContab();
   public void stopContab();
   public boolean estaAtivo(boolean remoto);
   public String GetCPU();
   public String getSystemName();

}
```

Quadro 1 Interface do Objeto *Computer*

Como pode ser visto no Quadro 1, o objeto *Computer* apresenta os seguintes métodos públicos:

- *getStatusContab*: identifica se a contabilização de recursos de CPU está ativa. Esta informação é utilizada pelo agente de contabilização para verificar se deve, ou não, realizar as tarefas de contabilização;
- *startContab*: inicializa a contabilização de recursos de CPU em uma base de dados relacional;
- *stopContab*: finaliza a contabilização de recursos de CPU;
- *estaAtivo*: verifica se o objeto está instanciado, ou não;
- *getCPU*: obtém a taxa de ociosidade de CPU da máquina em que está instanciado;
- *getSystemName*: obtêm as seguintes informações referentes ao sistema operacional da máquina: nome do *host*, número identificador da máquina e a versão do sistema operacional.

Após a construção da interface e implementação, pode-se construir um objeto-cliente para instanciar o objeto *Computer* na máquina desejada e realizar a invocação de métodos para este.

3.2 OBJETO *CLUSTER*

O objeto *Cluster* realiza a interface primária com a aplicação gerente. Através deste objeto, são efetuadas todas as operações necessárias a esta aplicação. O objeto referido instancia o objeto *Computer* em cada uma das máquinas do *cluster* e, através da invocação de métodos, realiza as operações de gerência desejadas.

Uma aplicação-gerente pode instanciar o objeto *Computer* diretamente em uma máquina e realizar as operações de gerência desejadas. No entanto, *applets* executados em *browsers Web*, só podem realizar a abertura de *sockets*, com a máquina que possui o servidor Web. Para solucionar esta limitação, pode-se instanciar o objeto *Cluster* na mesma máquina que executa o servidor Web e, através da invocações de métodos deste objeto, realizar o gerenciamento de todo o *Cluster* (Figura 3).

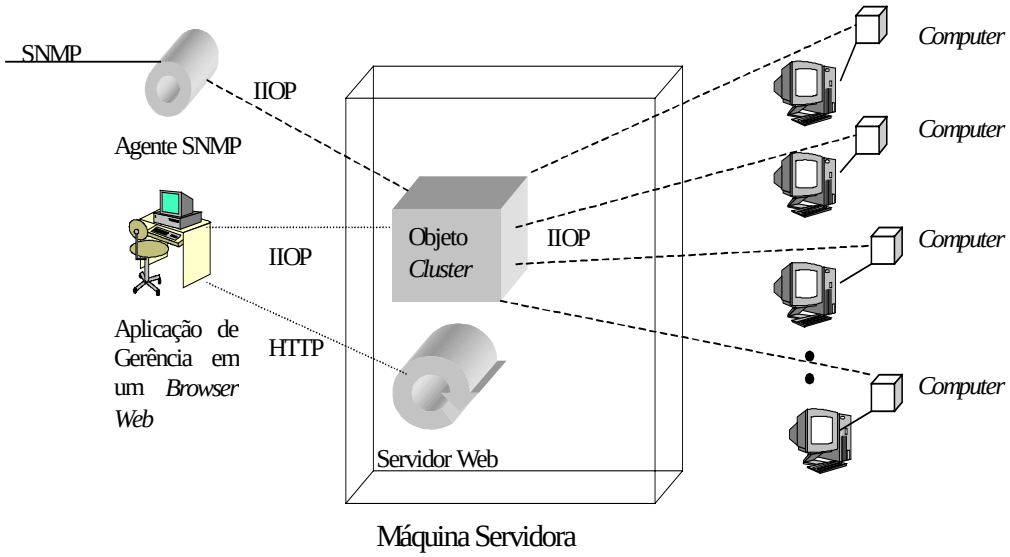

Figura 3 Objeto *Cluster*

Ao receber uma invocação de método, o objeto *Cluster* pode enviar mensagens para todas as máquinas do *cluster*, receber as respostas, processá-las e retornar os resultados encontrados. Por este motivo, para operações globais ou parciais ao *cluster*, ocorre uma diminuição do tráfego gerado e da carga de processamento relativa a aplicação-gerente. Pode-se exemplificar este fato na implementação de uma função de gerência para determinar as máquinas com as CPUs menos utilizadas no *cluster*. Neste caso, a aplicação-gerente pode enviar uma única mensagem requisitando a informação desejada. O objeto *Cluster*, ao receber a invocação de um método, envia uma mensagem para cada máquina, requisitando a taxa de ocupação de CPU. Uma vez de posse das respostas, o objeto *Cluster* processa os resultados e envia para o gerente uma lista das estações mais ociosas dentro do *cluster*. Neste caso, há uma diminuição da complexidade das operações relativas à aplicação-gerente, uma vez que esta envia apenas uma única mensagem, sem a necessidade de realizar qualquer processamento adicional, para determinar as máquinas mais ociosas. Esta característica é fundamental para implementação de aplicações de gerência, utilizando-se *browsers Web*. Sem o uso do objeto *Cluster*, a aplicação-gerente teria que enviar uma mensagem para cada máquina do *cluster*, receber as respostas e processar os resultados

.

3.2.1 IMPLEMENTAÇÃO DO OBJETO *CLUSTER*

Afim de apresentar a funcionalidade básica do objeto *Cluster* no gerenciamento de funções globais a um *cluster* de estações de trabalho, foi implementado um protótipo deste objeto. Na implementação deste protótipo, o objeto *Cluster* permite a contabilização de recursos, assim como, obter informações referentes ao sistema e CPUs das máquinas pertencentes ao ambiente. Este objeto mantém uma lista dos *hosts* pertencentes ao *Cluster* e obtém as informações de gerência através da invocação de métodos ao objeto *Computer*, instanciado nas diversas máquinas do *cluster*.

O objeto *Cluster* foi implementado em Java e, através de sua interface (Quadro 2), foram gerados os *Stubs* e *Skeletons*. Deste modo, este objeto pode ser instanciado por aplicações de gerência, utilizando o protocolo IIOP.

```
package agente;

import java.lang.Object;

public interface Cluster {

   public boolean getStatusContab();
   public void startContab();
   public void stopContab();
   public String getCPUStation(int Station);
   public String getSystemInfoStation(int Station);
   public String getClusterCPU();
   public String getClusterHostnames();
   public String getClusterSystemInfo();

}
```

Quadro 2 Interface do Objeto *Cluster*

No protótipo implementado, o objeto *Cluster* apresenta os seguintes métodos públicos:

- *getStatusContab*: retorna *true*, caso esteja ativa a contabilização de recursos de CPU. Este método é utilizado pelo agente de contabilização para verificar se deve ou não efetuar a contabilização de recursos de CPU, em uma base de dados relacional;
- *startContab*: inicia o contabilização de recursos de CPU;
- *stopContab*: finaliza a contabilização de recursos de CPU;
- *getCPUStation*: obtém a taxa de ociosidade de uma estação individual do *cluster*;
- *getSystemInfoStation*: obtém informações sobre o sistema de uma máquina individual no *cluster*;
- *getClusterCPU*: obtém um *string* contendo a taxa de ociosidade de todas as máquinas do *cluster* separadas por ";";
- *getClusterHostnames*: obtém um *string* com a lista de nomes de todas as máquinas do *cluster* separados por ";";
- *getClusterSystemInfo:* obtém informações sobre o sistema de todas as máquinas do *cluster*.

4 APLICAÇÃO DE GERÊNCIA CORBA

Para o gerenciamento das estações de trabalho pertencentes ao *cluster*, foi desenvolvida uma aplicação que pode ser distribuída para diversas máquinas da rede, utilizando a tecnologia *Web*. Esta aplicação visa a realização de experiências práticas com a utilização da arquitetura CORBA, da tecnologia WWW e da linguagem Java na implementação de sistemas de gerência distribuída.

A aplicação foi implementada como um *applet* Java (Figura 4). Em conseqüência deste fato, ela possui a característica de ser independente de plataforma e de poder ser utilizada, a partir de praticamente qualquer computador pertencente a Internet, sem a necessidade da instalação prévia de algum *software* de gerência.

CORBA foi utilizada para realizar a instanciação do objeto *Cluster*. Com a utilização da CORBA, a implementação do objeto *Cluster* é executada em um servidor e o *applet* necessita, apenas, de conter a definição da interface deste objeto. Por este motivo, a aplicação de gerência torna-se mais simples, contendo apenas o código necessário para realização da interface com o usuário. Outro ponto a ser observado, é que a implementação do objeto *Cluster* está sendo executada no servidor e não no *browser*. Em conseqüência deste fato, as operações realizadas pelo objeto *Cluster* não estão sujeitas às limitações de segurança impostas a *Applets.*

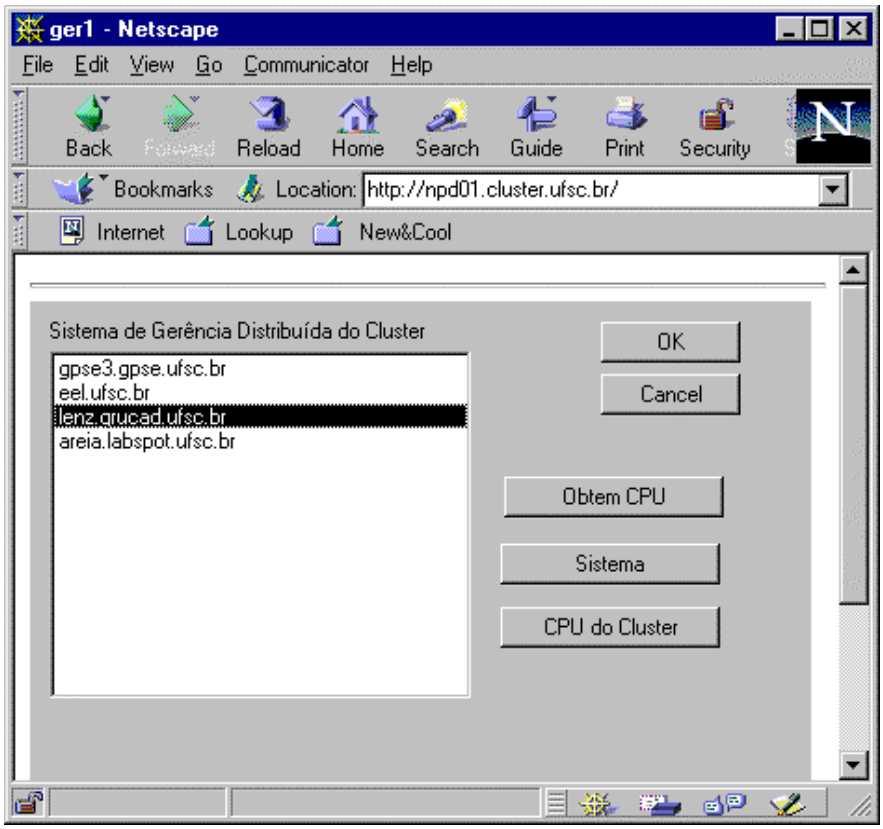

Figura 4 Janela Principal da Aplicação de Gerência

4.1 IMPLEMENTAÇÃO DE UM PROTÓTIPO DA APLICAÇÃO DE GERÊNCIA

Afim de apresentar um exemplo prático de uma aplicação de gerência CORBA, foi implementado um protótipo do sistema cuja finalidade é monitorar a taxa de ociosidade de CPU das máquinas do *cluster* e obter informações referentes ao sistema operacional de tais máquinas.

A fim de diminuir a complexidade do ambiente de estudo, os testes foram realizados utilizando apenas quatro estações pertencentes ao *cluster*. No entanto, todos os testes realizados podem, prontamente, abranger um número maior de estações.

A aplicação de gerência visa implantar a funcionalidade básica para a realização de operações de gerência globais ao *cluster*. Para exemplificar esta funcionalidade, na aplicação de gerência implementada neste projeto é possível realizar operações para obter a taxa de ociosidade de uma máquina individualmente ou de todas as estações do *cluster* simultaneamente. Ambas as operações exigem que a aplicação-gerente realize apenas uma única invocação de método ao objeto *Cluster*. Nesta aplicação, é possível, também, obter informações sobre o sistema operacional de todas as máquinas do ambiente, através do envio de uma única mensagem ao objeto *Cluster*.

As funções de gerência para o acompanhamento da taxa de ociosidade das máquinas do *cluster* podem ser muito úteis para um usuário observar quais as máquinas que possuem a menor utilização de CPU. Tais funções podem ser úteis, também, para um administrador realizar um estudo sobre o nível de utilização das máquinas e auxiliá-lo a decidir o momento necessário para realização de uma atualização destes equipamentos.

A função para a obtenção de informações sobre o sistema das máquinas do *cluster* pode auxiliar na aquisição de parâmetros como o *hostname* e o *hostid* dos equipamentos. Tais informações são necessárias, por exemplo, para a instalação de licenças de *softwares* nas máquinas. Caso o usuário tivesse que realizar esta tarefa manualmente, ele teria que se conectar a cada uma das estações do *cluster* e requisitar as informações desejadas. Esta operação poderia demandar grande quantidade de tempo.

Na implementação da aplicação de gerência, foram definidas somente funções para aquisição de informações não sigilosas do sistema. Esta limitação ocorre porque, neste trabalho, ainda não foram definidos procedimentos de autenticação e segurança para o acesso às funções de gerência.

Para a implementação deste protótipo foram utilizados os seguintes ambientes de desenvolvimento Java: o Microsoft J++ e o JDK 1.0.2.

5 AGENTE DE CONTABILIZAÇÃO

Para o armazenamento periódico de informações de contabilização, em uma base de dados relacional, foi implementado um Agente de Contabilização. Este agente é utilizado, neste trabalho, para o armazenamento periódico da taxa de ociosidade das máquinas do *cluster*. Estas informações podem ser muito úteis para realizar uma análise do perfil de ocupação das máquinas, durante um determinado período. Por exemplo, pode-se determinar quais os horários em que as máquinas estão mais ociosas, durante um determinado dia.

Quando executado, a primeira operação que é realizada pelo agente de contabilização é a instanciação do objeto *Cluster*. Neste caso, deve-se ressaltar que a aplicação de gerência e o agente de contabilização utilizam a mesma instância deste objeto. Por este motivo, se a aplicação de gerência alterar o valor de um atributo da instância utilizada pela aplicação de gerência, esta alteração será percebida, também, pelo agente de contabilização. Utilizando-se desta característica, o agente periodicamente verifica, junto ao objeto *Cluster*, se a função de contabilização está ativa e, caso esteja, realiza o armazenamento de informações em uma base de dados relacional. Dentro deste contexto, o processo-agente realiza basicamente as seguintes operações (Figura 5):

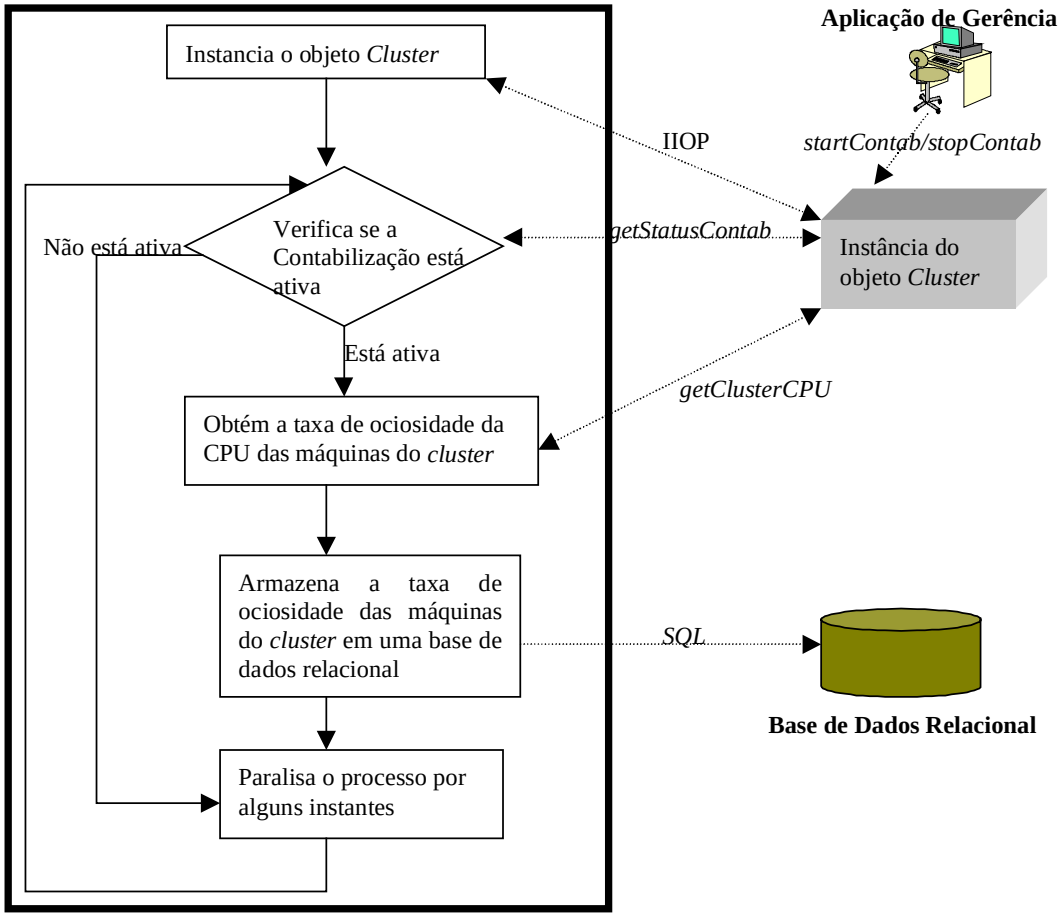

Figura 5 Funcionalidade do Agente de Contabilização

1. instancia o objeto *Cluster*;
2. realiza o invocação do método *getClusterCPU* do objeto *Cluster*, visando verificar se a função de contabilização de recursos de CPU está ativa. Esta função pode ser ativada por uma aplicação de gerência, através da invocação do método *startContab*, pertencente ao objeto *Cluster*. Para desativar esta função, a aplicação de gerência deve invocar o método *stopContab*. Esta interação entre o agente e a aplicação de gerência é possível porque ambos referenciam a mesma instância do objeto *Cluster*;
3. caso a função de contabilização esteja desativada, o processo-agente é paralisado por alguns instantes e realiza o passo 2 novamente;
4. se a função de contabilização estiver ativa, o processo-agente invoca o método *getClusterCPU* do objeto *Cluster*, visando obter a taxa de ociosidade de CPU de todas as máquinas gerenciadas;
5. as taxas de ociosidade são armazenadas em uma base de dados relacional, assim como, a data, a hora e o minuto em que foi realizada a aquisição dos dados;
6. o processo fica paralisado por alguns instantes e volta a executar o passo 2.

Um fato que deve ser observado é que este modelo de agente pode ser utilizado para a contabilização de qualquer tipo de informações de gerência. A implementação deste agente foi realizada utilizando a Java.

Caso seja necessário a realização da contabilização de recursos de uma única máquina, pode-se criar um agente que instancie o objeto *Computer* e execute esta função, seguindo a mesma funcionalidade apresentada pelo agente descrito anteriormente.

6 AGENTE SNMP

O agente SNMP, implementado neste projeto, apresenta uma estrutura bastante simples e tem a função de permitir a interação entre os protocolos SNMP e IIOP/CORBA (Figura 6). Este agente, ao receber primitivas *get-request*, instancia o objeto *Cluster*, envia as requisição desejadas e transmite uma PDU *get-response* (contendo os resultados obtidos) para a aplicação gerente. Deste modo, uma aplicação de gerência SNMP pode acessar objetos gerenciados que são instanciados através do protocolo IIOP/CORBA.

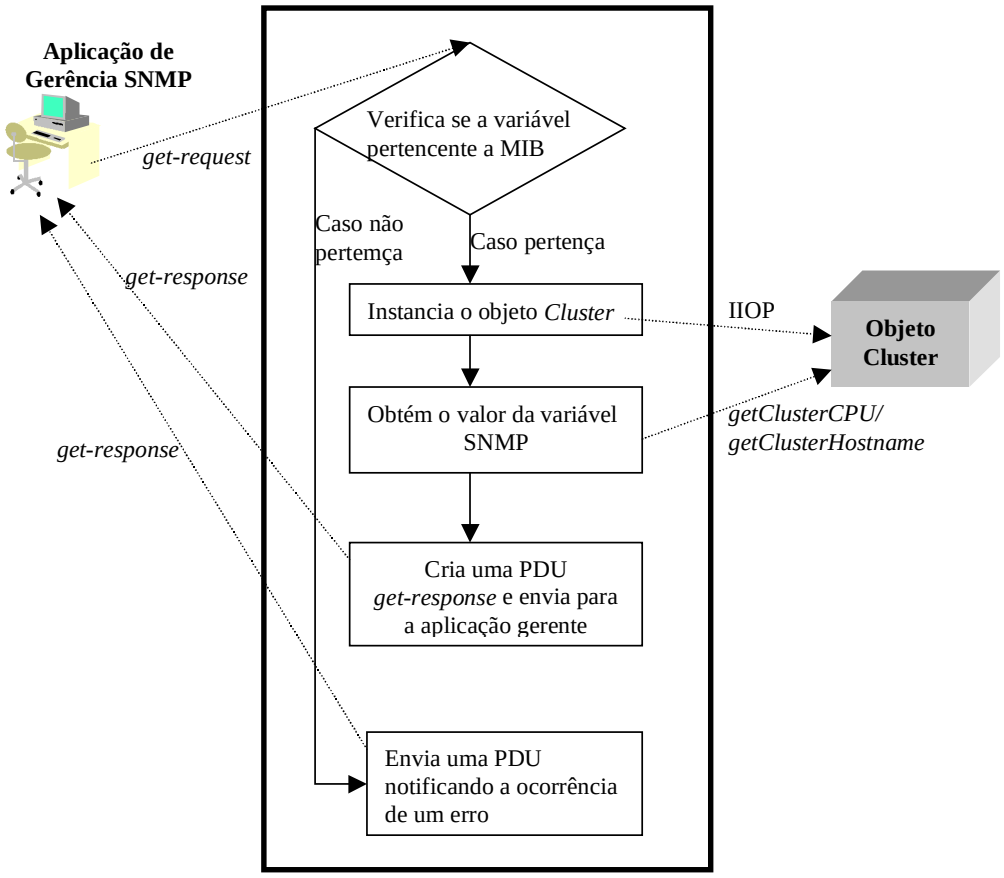

Figura 6 Funcionalidade do Agente SNMP

Para a implementação deste agente foi utilizado um modelo disponível na documentação da *Advent SNMP API* (*http://www.adventnet.com*). Este modelo foi adaptado para permitir a instanciação de objetos gerenciados, através da CORBA, e viabilizar a utilização de uma MIB

própria para o *cluster*. O agente SNMP é capaz de retornar informações sobre o nome e a taxa de ociosidade da CPU das máquinas que compõem o *Cluster*.

7 CONCLUSÃO

Através deste projeto, pôde-se obter uma boa experiência na utilização do CORBA, WWW e Java para o gerenciamento de redes, permitindo a construção de um sistema para o gerenciamento de um *cluster* de estações que é integrável com aplicações de gerência SNMP. O protótipo implementado neste projeto define uma estrutura simplificada de gerenciamento em que os objetos gerenciados fornecem informações sobre o sistema e CPU das máquinas pertencentes ao *cluster*. No entanto, as experiências adquiridas com este sistema compõem a base para a implantação de um sistema de gerência completo para o *cluster* de estações.

Com as experiência adquiridas neste projeto, pôde-se definir um mecanismo de gerência em que os objetos gerenciados, ORBs, aplicações de gerência e agentes são implementados em Java. Com isto, obtém-se um mecanismo de gerenciamento e instanciação de objetos portável para qualquer plataforma que possua um interpretador Java instalado.

Um dos experimentos realizados neste projeto, foi a utilização da classe *Runtime* da Java para permitir que objetos instanciados (utilizando a CORBA) possam invocar comandos de um *shell* UNIX. Através deste mecanismo, é possível a execução de praticamente qualquer comando que um usuário poderia invocar localmente a uma máquina. Em conseqüência disto, foi criada a infra-estrutura de gerência necessária para que um usuário possa obter informações ou emitir comandos para as máquinas do *cluster*, a partir de um *browser Web*, como se estivesse conectado localmente a uma máquina.

Uma das metas a serem alcançadas com este projeto é a definição de novas funções para o sistema de gerência, que permitam a um usuário, através da interação com um *applet* Java, realizar tarefas como: instalação automática de *softwares* em todas as máquinas do *cluster*, remoção de arquivos de *logs* antigos, gerenciamento de áreas de disco, entre outras. No entanto, a fim de viabilizar a execução de tais tarefas, deve-se primeiramente implementar mecanismos de autenticação e criptografia que permitem a transmissão de mensagens de gerenciamento de modo seguro entre os objetos. Em virtude disto, um dos passos a serem seguidos na continuidade deste projeto é a definição de mecanismos de segurança para o gerenciamento do *cluster*.

Uma das dificuldades encontradas neste projeto foi o mapeamento de funções de gerência entre o protocolo SNMP e o IIOP/CORBA. Parte destas dificuldades deve-se ao fato de que os recursos oferecidos pelo SNMP são limitados quando comparados a potencialidades do CORBA. Em virtude disto, definiu-se que nas próximas versões deste sistema somente serão mapeadas para o protocolo SNMP, as funções que apresentarem maior relevância para o gerenciamento do sistema.

8 BIBLIOGRAFIA

1. BARILLAUD, Frank; DERI, Luca; FERIDUN, Metin. Network management using internet technologies. *Fifth IFIP/IEEE International Symposium on Integrated Network Management*, San Diego, CA, USA, p. 61-70, 1997.

2. BAROTTO, André M.; SOUZA, Adriano de; WESTPHALL, Carlos. B. Distributed Network Management using SNMP, JAVA, WWW and CORBA. *Journal of Network and Systems Management*. Plenum Publishing Corporation. Meddletown, USA, 1999. (Accepted with transfer of copyright).

3. BERNHARDT, Martin. *Design and implementation of a web-based tool for ATM connection management*. Stuttgart, Aug. 1996. Master's Thesis – Department of Computer Science, University of Stuttgart.

4. *Bojanbles*. IBM, 1997, http://www.alphaWorks.ibm.com/formula

5. EVANS, Eric; ROGERS, Daniel. Using java applets and CORBA for mult-user distributed applications. *IEEE Internet Computing*, May/June. 1997, http://computer.org/internet/.

6. GUNTHER, Oliver; MULLER, Rudolf; SCHIMDT, Peter. MMM: a web-based system for sharing statistical computing modules. *IEEE* Computing; May/June. 1997, http://comuter.org/internet/.

7. HERMAN, James.Web-Based net management is coming. *Data Communications International*, New York, Oct. 1997, p. 138-141..

8. KRULWICH, Bruce. Automating the internet: Agents as User Surrogates. *IEEE* Computing; July/Aug. 1997, http://comuter.org/internet/.

9. LARSEN, Amy. The next web wave: network management. *Data Communications*, p. 31-34, jan. 1996.

10. LAPPINEN, Mika; PULKKINEN, Pekka; RAUTIAINEN, Aapo. Java and CORBA Based Network Management. *IEEE Computer*, June. 1997, http://dlib.computer.org/.

11. MILLER, Mark A. *Managing Intertworks with SNMP:* the definitive guide to simple network mangement protocol, SNMPv2, RMON, and RMON2. NewYork, M&T, 1997.

12. MASTON, M. C. Using the world wide web and java for network service management. *Fifth IFIP/IEEE International Symposium on Integrated Network Management*, San Diego, CA, USA, p. 71-84, 1997.

13. ORFALI, Robert; HARKEY, Dan. *Client/Server programming whith JAVA and CORBA*. New York : John Wiley & Sons, 1997.

14. ORFALI, Robert; HARKEY, Dan; EDWARDS, Jeri. *The essential distributed objects survival guide*. NewYork : John Wiley & Sons, 1996.

15. OTTE, Randy; PAUL, Patrick; ROY, Mark.*Understanding CORBA*. New Jersey: Prentice Hall, 1996

16. REED, B. Distributed systems management on the web. *Fifth IFIP/IEEE International Symposium on Integrated Network Management*, San Diego, CA, USA, p. 85-95, 1997.