Escola Politécinica – USP

Programa de Pós Graduação

Estudo sobre Aplicações de Gerenciamento de Redes Open Source existentes e suas potencialidades

PCS5742 – Gerenciamento de Rede

Prof. Denis Gabos

Alunos:

Alexandre Cassiano

Cláudio Penásio

Wagner L. S. Pereira

Setembro/2004

Sumário

1 Introdução

Com a evolução das tecnologias de redes e com a expansão das redes empresariais o gerenciamento da rede assumiu um papel preponderante para manter a vitalidade das mesmas.

A princípio a necessidade de gerenciamento apareceu nas operadoras de telecomunicações como parte da operação técnica e progrediu para a operação dos negócios.

Com a expansão das redes corporativas apareceram no mercado várias suites de gerencia de redes. Estas suites foram ficando cada vez mais completas e abrangentes, porém são acessíveis apenas para grandes corporações, deixando à margem pequenas e médias redes.

A adoção do código aberto potencializou uma série de novas soluções e aplicações, criando comunidades de desenvolvimento para os mais variados campos da Tecnologia da Informação.

> Um dos campos que se desenvolveu na área de código aberto foi o de gerência de redes. Existem várias iniciativas neste campo, algumas mais evoluídas e outras nem tanto. Nosso objetivo é avaliar algumas destas soluções e estudar os recursos que oferecem e o grau de maturidade da solução.

2 Overview

As soluções comerciais disponíveis são várias, algumas muito específicas e destinadas a um equipamentos do próprio fabricante (p.e. Cisco ..), outras específicas para uma determinada função (análise de tráfego p.e.). As soluções mais presentes em grandes corporações podem ser resumidas em 3 grandes Suites: o OpenView da HP, o Tívoli da IBM e o TNG da CA.

Dependendo das funcionalidades e do tamanho da rede, uma solução para gerenciamento de rede de um dos top 3 pode atingir a cifras na ordem de centenas de milhares de dólares.

Pelo lado do código aberto, várias grupos foram criados para desenvolvimento de aplicações de gerencia de rede. O MRTG é uma ferramenta de monitoração de redes muito utilizada por profissionais de rede e foi uma das primeiras iniciativas neste sentido.

Com a criação do padrão WBEM foram criados grupos para implementar soluções baseadas nesta tecnologia. Surgiram então o OpenWBEM e OpenPegasus. O

OpenPegasus está sob a égide do The Open Group que possui outras iniciativas de desenvolvimento de código aberto. Fazem parte do grupo as HP, IBM e SUN entre outras.

O OpenWBEM foi criado para desenvolvimento de da implementação WBEM em código aberto, assim como no OpenPegasus os principais participantes são grandes empresas interessadas em uma plataforma aberta de gerenciamento. Fazem parte deste grupo a a Novell e a SCO.

Falar sobre o Nagios, JMX e outros ...

Para efeito de nosso estudo selecionamos algumas implementações que estão mais difundidas. Desta forma vamos estudar o Nagios, o JMX e o OpenPegasus.

3 Estudo das Soluções

3.1 Nagios™

3.1.1 Descrição da Solução

Nagios™ é um sistema de monitoramento de rede, ele monitora serviços e hosts de acordo com a especificação do usuário, alertando-o tanto nas situações boas como nas ruíns.

3.1.2 Principais Recursos

- Monitoramento de serviços de rede (SMTP,POP3, NNTP, PING, etc.);Monitoramento dos recursos dos hosts (processos em execução, utilização de disco, etc);
- Design simplificado de plugins, que permite aos usuários um desenvolvimento facilitado a seus próprios plugins de monitoramento;
- Checagem de serviços paralelizados;
- Habilidade para definir hierarquia de host de rede, através de um host “pai”, permitindo detecção e distinção entre hosts que estão desligados e inalcaçáveis;
- Emite notificações via e-mail, pager ou outro método definido pelo usuário quando ocorre um problema com um host ou serviço;
- Permite a definição de ações pro ativas quando da ocorrência de determinados eventos nos hosts ou serviços;

- Rotação de arquivo de log automática;
- Suporte para implementação de monitoramento redundante de hosts;
- Interface web opcional para visualizar o status da rede,notificações e histórico de problemas, arquivos de logs e etc.

3.1.3 Portabilidade

O Nagios™ roda sobre o Sistema Operacional Linux (Todos os sabores) e em sistemas variantes do Unix como os BSD's de Berkley. O requisito é que o sistema (Linux ou Unix) possua um compilador C, e evidentemente o TCP/IP configurado, pois é a base de funcionamento do Nagios™ .

Caso você deseje utilizá-lo no modo CGI (Common Gateway Interface), será necessário um servidor web (Preferencialmente o Apache) com suporte a biblioteca gd versão 1.6.3 ou mais nova.

3.1.4 Licença

Nagios™ é licenciado sob os termos da Licença *GNU General Public License Version 2* publicada pela *Free Software Foundation*. Isto lhe dá permissão legal para copiar, distribuir e/ou modificar o Nagios sob certas condições. Leia o arquivo 'LICENSE' na distribuição do Nagios ou leia a versão on-line da licença para maiores detalhes. Nagios™ não é provido de nenhuma garantia de nenhum tipo, incluindo a garantia de design, comercialização, e aptidão para uma determinada proposta.

3.1.5 Como Instalar

A Instalação do Nagios pode ser realizada basicamente de duas formas principais, sendo a primeira através de um pacote no formato RPM (caso você esteja usando Red Hat Linux 7.3 até o 9.0 ou Red Hat Enterprise Linux 2,1 ou 3.0 ou ainda o Fedora Core 1.0 ou 2.0). Ou através de um pacote TAR, neste caso não importa a Distribuição, pois você mesmo vai compilá-lo.

- RPM

Para instalar o pacote RPM digite:

rpm -ivh ***nagios-1.2-0.rh90.dag.i386.rpm***

- TAR

Para desempacotar e descompactar o pacote tar digite:

tar xvzf nagios-1.2.tar.gz

Após este passo será automaticamente criado no diretório corrente o diretório nagios-1.2, sendo que dentro desta haverá todo o núcleo da distribuição do nagios.

Para criar o diretório de instalação do Nagios no sistema digite:

mkdir /usr/local/nagios

Você provavelmente irá querer rodar o Nágios como um usuário normal do sistema, para isso crie o usuário e grupo “nagios”, para isso digite:

adduser nagios

Dependendo do sistema que você estiver rodando talvez você tenha que criar o grupo manualmente.

Rodando o script de configuração:

./configure –prefix=/usr/local/nagios –with-cgiurl=/nagios/cgi-bin –with-htmurl=/nagios --with-nagios-user=nagios --withnagios-grp=nagios

Compilando os binários:

make all

instalando os binários e arquivos html

make install

Instalando o script de inicialização /etc/rc.d/init.d/nagios

make install-init

Dependendo do sistema que você utiliza, talvez você tenha que editar o script de inicialização para corrigir paths, etc.

A estrutura de arquivos e diretórios do Nagios fica da seguinte forma:

/usr/local/nagios/bin (Núcleo de Programas Nagios)

/usr/local/nagios/etc (Main, resource, object, e CGI arquivos de configuração devem ser colocados aqui)

/usr/local/nagios/share (arquivos HTML - para interface web e documentação on-line)

/usr/local/nagios/var (Diretório vazio para arquivos de log)

3.1.6 Detalhamento das Funcionalidades

O Nagios possui uma arquitetura totalmente baseada em plugins, podemos afirmar com total segurança que o Nágios é inútil sem os plugins. Abaixo detalharemos algumas definições e o princípio de funcionamento.

1. Plugins: são arquivos executáveis ou scripts escritos geralmente em Perl e shell script, que são rodados para verificar um status, um host ou serviço. Os plugins são vitais, pois sem eles você não consegue monitorar nada.

   Para baixar os plugins do Nagios é só ir no site do projeto **http://sourceforge.net/projects/nagiosplug/**.

2. Addons: são adendos ao Nágios que funcionam na forma de Daemons e plugins, no site do Nagios existem vários addons para serem baixados, os dois principais addons são:

   1. nrpe - Daemon e plugin para executar plugins no hosts remoto

      Este addon foi projetado para prover uma forma de executar plugins em um host remoto. O plugin check_nrpe roda no host Nagios e é usado para enviar a requisição de execução do plugin para o agente nrpe no host remoto. O agente nrpe irá rodar um plugin apropriado no host remoto e retornará a saída do plugin e o código para o check_nrpe no host Nagios. O agente nrpe pode rodar como um daemon standalone ou um serviço sobre o inetd.

   2. nsca - Daemon e programa cliente para transmissão de resultados através da rede.

      Este addon permite o envio de resultados de checagens de serviço passivo no host remoto para um monitoramento central onde o Nagios estiver rodando. O cliente pode ser usedo como um programa standalone ou pode ser integrado com um servidor Nagios remoto que roda o comando ocsp para configurar um ambiente de monitoramento remoto. A comunicação entre o cliente e o daemon pode ser encryptada através de vários algorítmos (DES, 3DES, CAST, xTEA, Twofish, LOKI97,

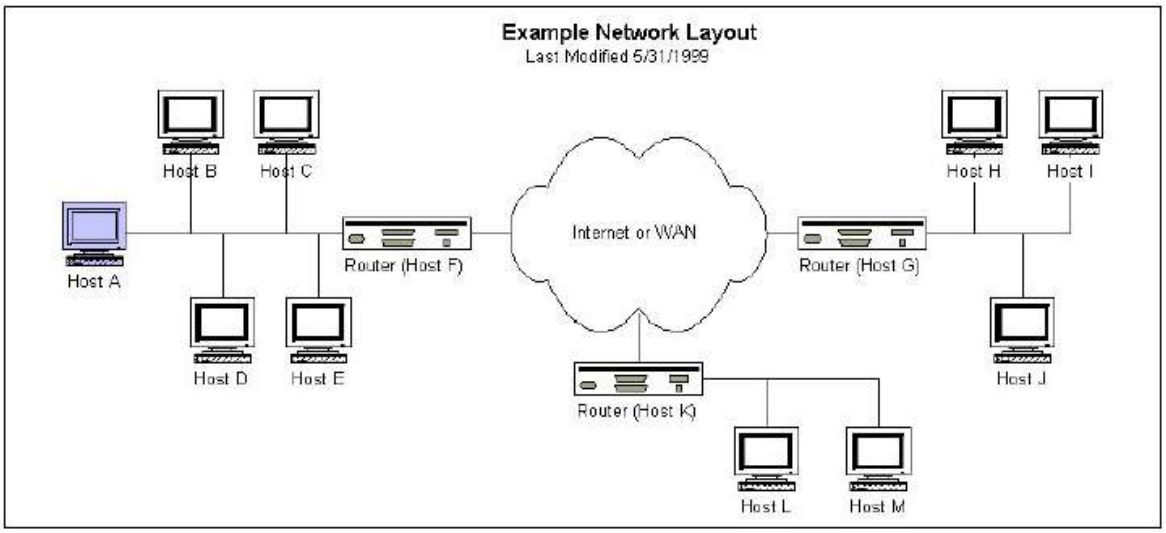
RJINDAEL, SERPENT, GOST, SAFER/SAFER+, etc.) se você tiver a biblioteca mcrypt no seu sistema.

3. Determinando o Status e o alcance dos hosts da rede

   A principal proposta do Nagios é monitorar serviços rodando em um host fisico ou dispositivo na rede, bem como monitorar o host . Embora seja intuitivo pensar que quando um host ou dispositivo “cai”, todos os serviços hospedados ali caem também. Similarmente, se um host torna-se inalcançável, o Nagios não poderá monitorar os serviços associados a este host. O Nagio reconhece este fato e tenta checar através de um cenário onde o problema seria no serviço. Sempre que a checagem de um serviço retorna um status de nível “non-OK”, o Nagios irá tentar checar e ver se o host que o serviço está rodando está “vivo”. Normalmente isso pode ser feito “pingando” o host e verificando se alguma resposta é recebida. Se a checagem de host retorna um estado “non-OK”, o Nagios assume que há um problema coim o host. Nesta situação o Nagios irá “silenciar” todos os alertas de serviços rodando neste host e apenas notificar os contatos adequados para o caso de host parado ou inalcançável. Se o comando de check de host retorna estado OK, o Nagios reconhece que o host está “vivo” e irá enviar um alerta para o serviço que está se comportando mau.

4. Hosts Locais

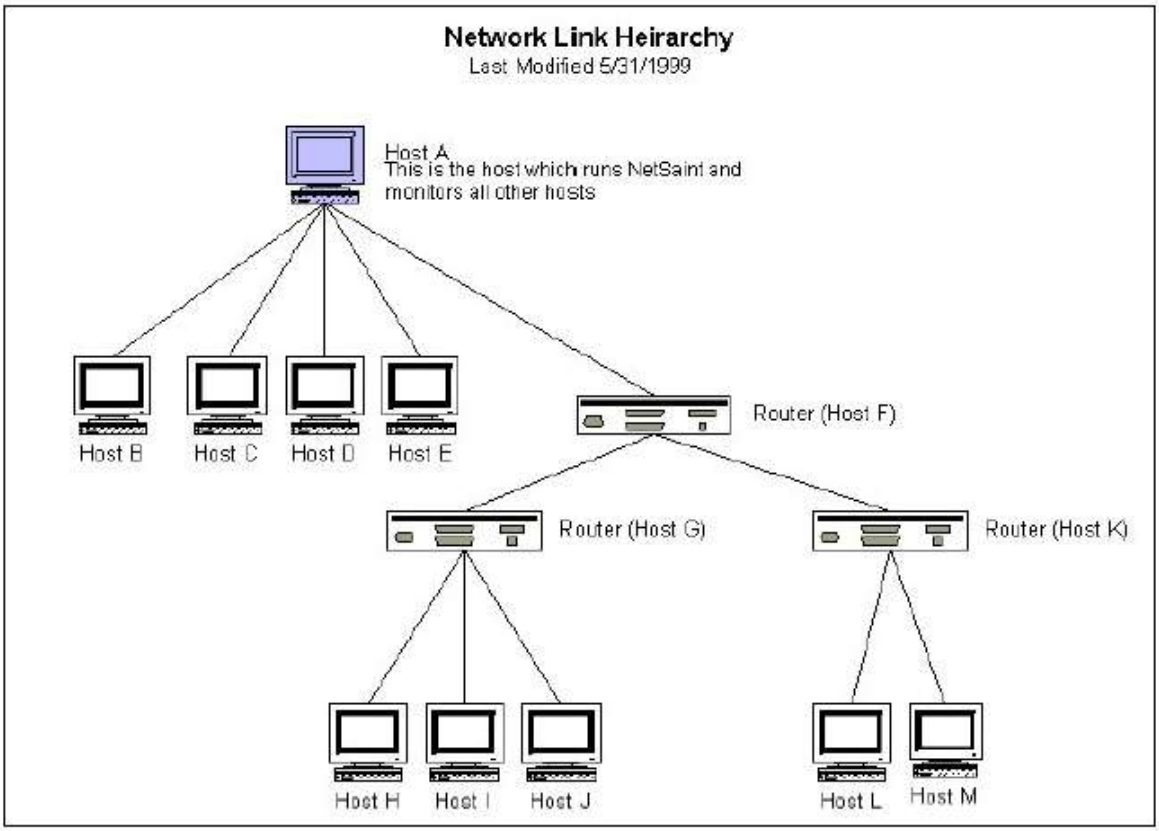
   Hosts “locais” são hosts que estão no mesmo segmento de rede que o host que está rodando o Nagios, sem roteadores ou firewalls entre eles. A Figura abaixo mostra um exemplo deste lay-out de rede. Host A está rodando o Nagios e monitorando todos os outros hosts e roteadores do diagrama. Os hosts B, C, D, E e F são todos considerados hosts locais em relação ao host. A opção “<parent_hosts>” na definição de host, para um host “local”deve ser deixada em branco.



5. Host Remoto

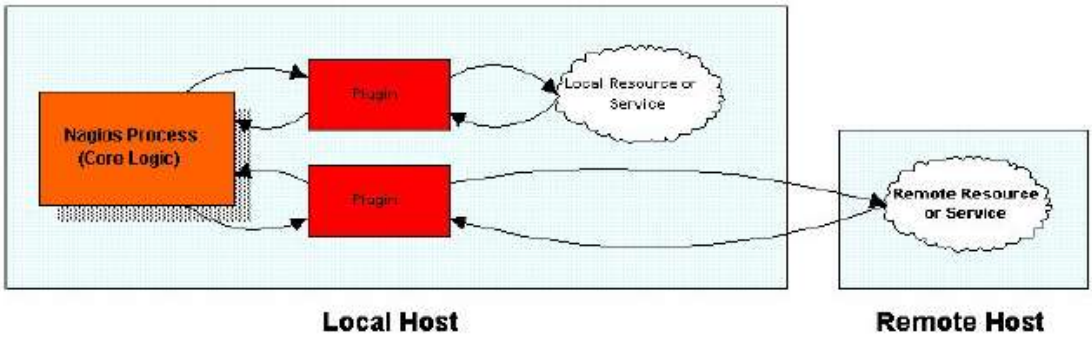
Hosts “Remotos” são hosts que residem em um segmento de rede diferente do que o Nagios está rodando. Na figura acima, os hosts G, H, I, J, K, L e M são todos considerados remotos em relação ao host. O aviso que alguns hosts estão “longe” de outros. O Host H,I e J está a um salto de distância do Host A que está a um salto do Host G (roteador). Para esta visão podemos construir uma árvore de dependências de hosts como visto na Figura abaixo. Este diagrama é de grande importância na para entendermos a configuração dos hosts no Nagios.

A opção “<parent_hosts>” na definição de host para um host “remoto” deverá ser um nome de hosts curto diretamente acima no diagrama (como mostrado abaixo). Por exemplo, o “parent Host” para o host H deverá ser o Host G. O o “parent Host” para o host G é o Host F. O Host F não tem “parent Host” , pois ele está no mesmo segmento de rede que o host A, logo ele é um host “Local”.



6. Como Funcionam os Plugins

Diferentemente de outras ferramentas de monitoramento, o Nagios não possui nenhum mecanismo interno de para checar o status dos serviços, hosts e etc. Em vez disso, Nagios utiliza programas externos chamados plugins para fazer todo o trabalho. O Nagios executará um plugin sempre que haver uma necessidade de checar um serviço ou um host de ser monitorado. O plugin faz alguma coisa para realizar a checagem e simplesmente retorna os resultados ao Nagios. O Nágios irá processar os resultados recebidos do plugin e tomar qualquer ação necessária (rodando eventos, enviando notificações, etc). A imagem abaixo mostra como os plugins são separados do núcleo do Nágios.

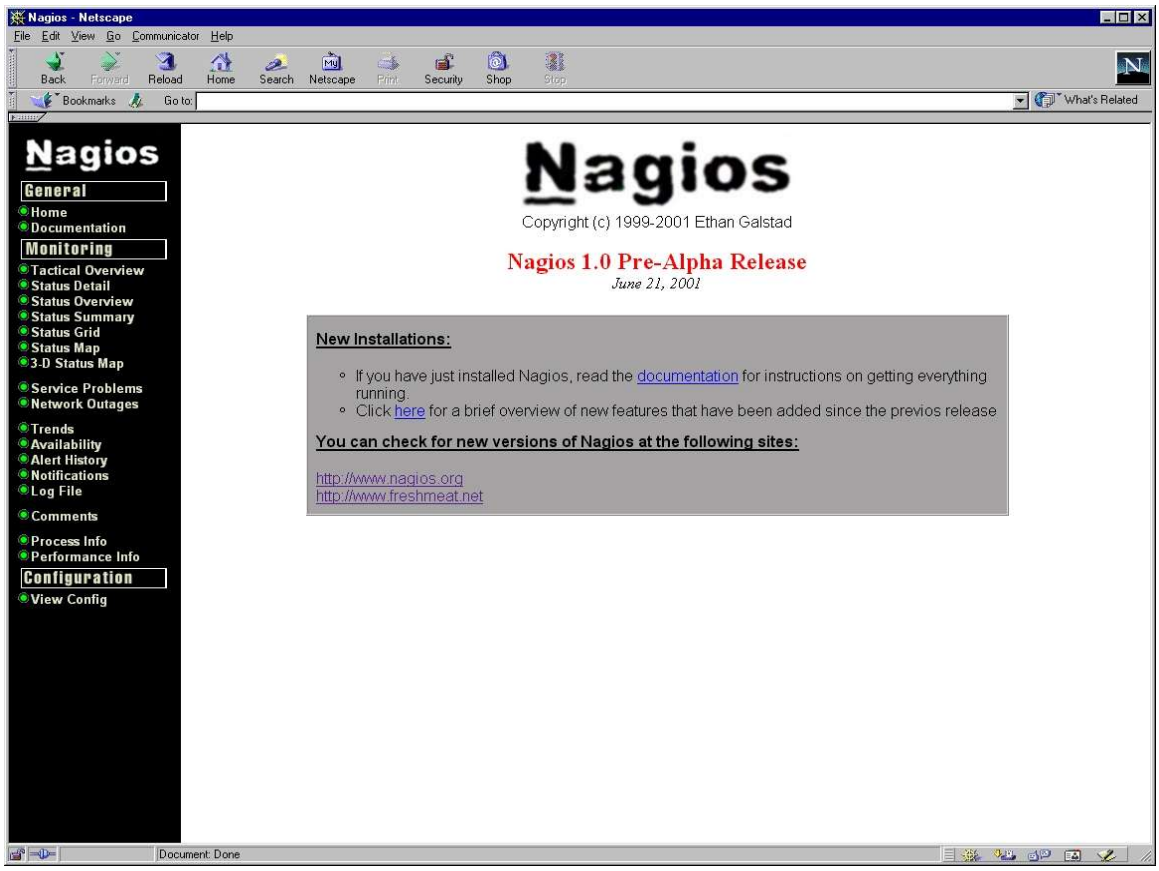


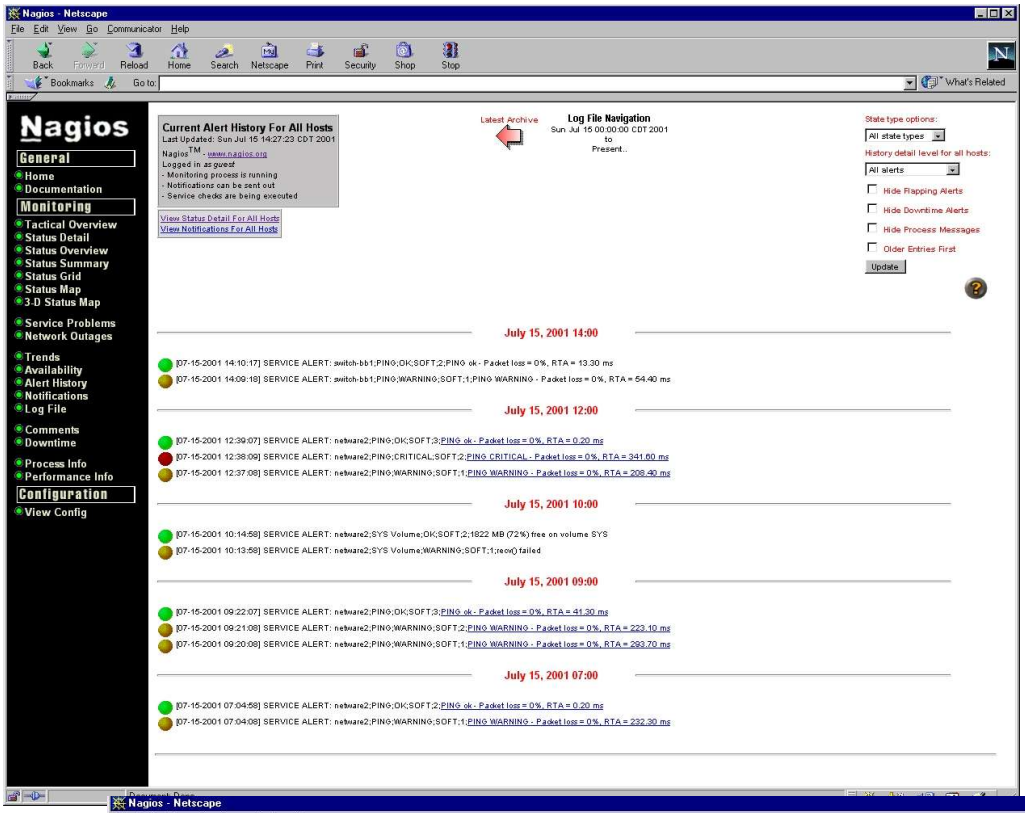
Uma boa coisa sobre a arquitetura de plugins é que você pode monitorar características totalmente isoladas. Já estão pronyos uma série de plugins que foram criados para monitorar recursos básicos como carga de processador, uso de disco, ping, taxas e etc.

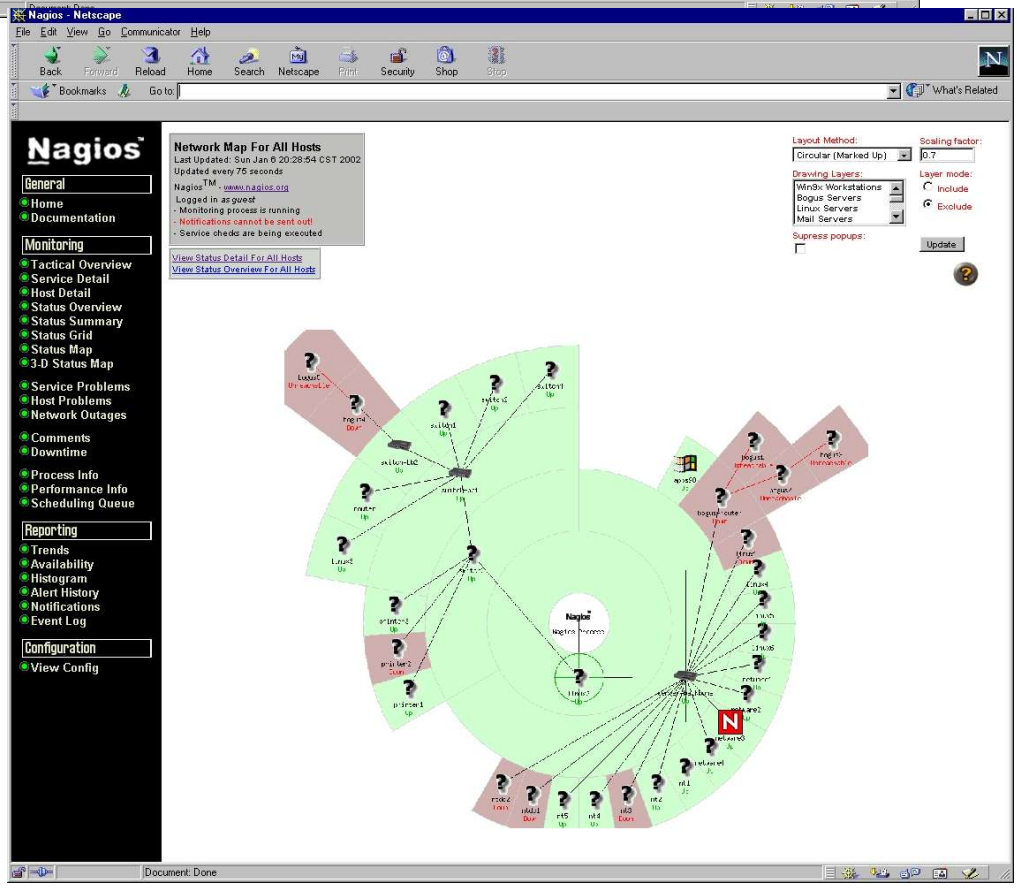
Um ponto negativo na arquitetura de plugins é que o Nagios não tem a menor idéia a respeito do que está sendo monitirado, todo o tratamento de dados é realizado por plugins, ou também pode ser realizado por aplicações externas.

7. Interface web

   Para obter um gerenciamento mais eficaz o Nagios possui uma interface web, que através dos mais variados plugins pode fornecer uma visão ampla e confortável dos recursos monitorados. Demonstramos abaixo alguns screenshots:




Nagios - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Shop Stop
Bookmarks Go to: What's Related
Nagios
General
Home
Documentation
Monitoring
Tactical Overview
Status Detail
Status Overview
Status Summary
Status Grid
Status Map
3-D Status Map
Service Problems
Network Outages
Trends
Availability
Alert History
Notifications
Log File
Comments
Downtime
Process Info
Performance Info
Configuration
View Config
Current Alert History For All Hosts
Last Updated: Sun Jul 15 14:27:23 CDT 2001
Nagios™ - www.nagios.org
Logged in as guest
- Monitoring process is running
- Notifications can be sent out
- Service checks are being executed
View Status Detail For All Hosts
View Notifications For All Hosts
Latest Archive
Log File Navigation
Sun Jul 15 00:00:00 CDT 2001
to
Present..
State type options:
All state types
History detail level for all hosts:
All alerts
Hide Flapping Alerts
Hide Downtime Alerts
Hide Process Messages
Older Entries First
Update
July 15, 2001 14:00
[07-15-2001 14:10:17] SERVICE ALERT: switch-bb1;PING;OK;SOFT;2;PING ok - Packet loss = 0%, RTA = 13.30 ms
[07-15-2001 14:09:18] SERVICE ALERT: switch-bb1;PING;WARNING;SOFT;1;PING WARNING - Packet loss = 0%, RTA = 54.40 ms
July 15, 2001 12:00
[07-15-2001 12:39:07] SERVICE ALERT: netware2;PING;OK;SOFT;3;PING ok - Packet loss = 0%, RTA = 0.20 ms
[07-15-2001 12:38:09] SERVICE ALERT: netware2;PING;CRITICAL;SOFT;2;PING CRITICAL - Packet loss = 0%, RTA = 341.60 ms
[07-15-2001 12:37:08] SERVICE ALERT: netware2;PING;WARNING;SOFT;1;PING WARNING - Packet loss = 0%, RTA = 208.40 ms
July 15, 2001 10:00
[07-15-2001 10:14:58] SERVICE ALERT: netware2;SYS Volume;OK;SOFT;2;1822 MB (72%) free on volume SYS
[07-15-2001 10:13:58] SERVICE ALERT: netware2;SYS Volume;WARNING;SOFT;1;(null) failed
July 15, 2001 09:00
[07-15-2001 09:22:07] SERVICE ALERT: netware2;PING;OK;SOFT;3;PING ok - Packet loss = 0%, RTA = 41.30 ms
[07-15-2001 09:21:08] SERVICE ALERT: netware2;PING;WARNING;SOFT;2;PING WARNING - Packet loss = 0%, RTA = 223.10 ms
[07-15-2001 09:20:08] SERVICE ALERT: netware2;PING;WARNING;SOFT;1;PING WARNING - Packet loss = 0%, RTA = 293.70 ms
July 15, 2001 07:00
[07-15-2001 07:04:58] SERVICE ALERT: netware2;PING;OK;SOFT;2;PING ok - Packet loss = 0%, RTA = 0.20 ms
[07-15-2001 07:04:08] SERVICE ALERT: netware2;PING;WARNING;SOFT;1;PING WARNING - Packet loss = 0%, RTA = 232.30 ms


Nagios - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Shop Stop
Bookmarks Go to: What's Related
Nagios
General
Home
Documentation
Monitoring
Tactical Overview
Service Detail
Host Detail
Status Overview
Status Summary
Status Grid
Status Map
3-D Status Map
Service Problems
Host Problems
Network Outages
Comments
Downtime
Process Info
Performance Info
Scheduling Queue
Reporting
Trends
Availability
Histogram
Alert History
Notifications
Event Log
Configuration
View Config
Network Map For All Hosts
Last Updated: Sun Jan 6 20:28:54 CST 2002
Updated every 75 seconds
Nagios™ - www.nagios.org
Logged in as guest
- Monitoring process is running
- Notifications cannot be sent out!
- Service checks are being executed
View Status Detail For All Hosts
View Status Overview For All Hosts
Layout Method:
Circular (Marked Up)
Scaling factor:
0.7
Drawing Layers:
Win9x Workstations
Bogus Servers
Linux Servers
Mail Servers
Layer mode:
Include
Exclude
Supress popups:
Update
Document: Done

3.2 O JMX da Sun

O JMX [JMXWP 1999] foi criado por um consórcio de empresas da área de sistemas de gerência para suprir os requisitos de mercado para o gerenciamento dinâmico de recursos e de sistemas através da tecnologia Java. O JMX oferece todo o suporte requerido pelos desenvolvedores de soluções de gerência. Um estudo aprofundado da especificação do JMX, dirigido especificamente ao mundo das telecomunicações, é importante para que esta tecnologia seja validada antes de que soluções completas para sistemas de gerência TMN sejam oferecidas ao mercado.

Através do mapeamento de um modelo de informação de gerência para os JavaBeans utilizados pela arquitetura especificada pelo JMX, cria-se a infraestrutura necessária para o desenvolvimento de um sistema de gerência TMN baseado na tecnologia Java.

Atendendo a esta exigência de mercado, a Sun desenvolveu uma plataforma de desenvolvimento de aplicações de gerência baseada na tecnologia especificada pelo JMX: o Java DMK. O Java DMK [JDMKWP 1998] [JDMK 1999] é a primeira solução integrada baseada na tecnologia Java para o desenvolvimento de soluções dinâmicas de gerência e para a distribuição natural do gerenciamento entre os dispositivos da rede. Esta plataforma de desenvolvimento baseia-se em componentes de software JavaBeans e possibilita, de forma eficiente e independente, a criação de gerentes e de agentes Java para sistemas, redes e serviços.

.

3.2.1 A Estrutura do JMX da Sun

As extensões de gerência Java (JMX) [JMX 1999] definem a arquitetura, os padrões de especificação, as APIs (Application Program Interfaces) e serviços para aplicações de gerenciamento de redes baseadas na tecnologia Java. O JMX provê aos desenvolvedores suporte para a criação de agentes Java e para a implementação de aplicações distribuídas de gerência, assim como APIs de

integração destas soluções com tecnologias de gerência padrões atualmente utilizados pelo mercado.

A arquitetura do JMX é dividida em três níveis: nível instrumentação, nível agente e nível gerente. A Figura 1 a seguir apresenta a forma com que estes componentes se relacionam entre si dentro dos três níveis:

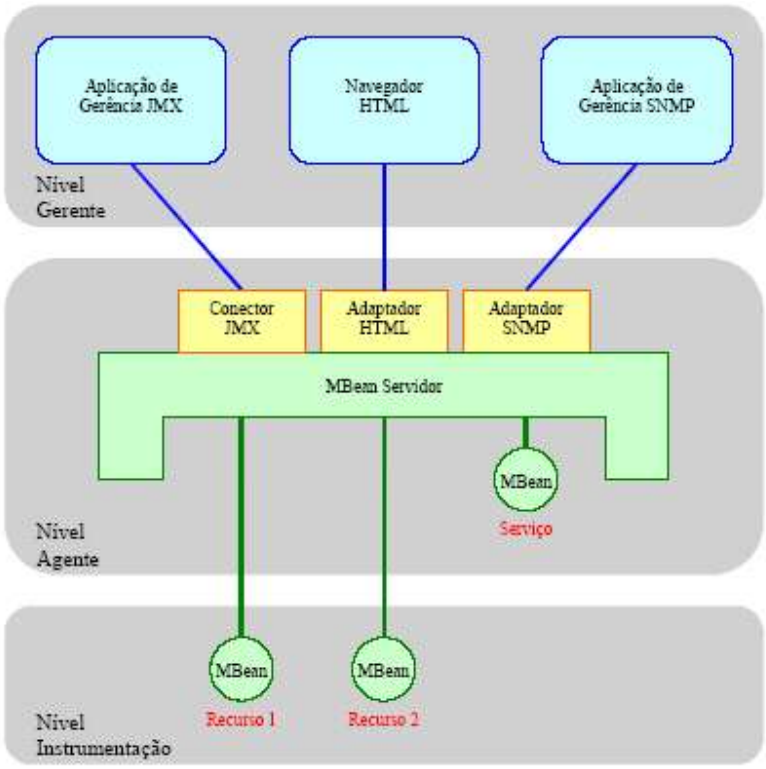


Figura 1: Relacionamento entre os componentes da arquitetura do JMX

3.2.1.1 Nível Instrumentação

O nível instrumentação corresponde a uma especificação da forma com que os recursos gerenciáveis devem ser implementados usando os JavaBeans. Os recursos gerenciáveis podem ser uma aplicação, um serviço, um dispositivo ou um usuário, desde que implementados de acordo com a especificação do JMX.

A instrumentação de um determinado recurso é provida por um ou mais Java Beans de gerência, os MBeans, que são gerenciados através do nível agente. Os Mbeans são projetados para serem flexíveis, simples e fáceis de serem implementados. Desenvolvedores de aplicações, de serviços ou até mesmo de dispositivos, através desta metodologia podem tornar seus produtos gerenciáveis de uma forma padrão sem que para isto tenham que investir e dominar sistemas de gerenciamento complexos.

O nível instrumentação também define um mecanismo de notificação, o qual permite que os MBeans gerem e propaguem eventos de notificações para componentes de outros níveis.

Mbeans

Os objetos Java que implementam as características e funcionalidades de um determinado recurso são denominados de JavaBeans5 de gerenciamento, ou abreviadamente, MBeans. Os MBeans devem seguir rigorosamente os padrões de especificação e as interfaces definidas pela especificação do JMX, de forma

que se possam gerenciar os recursos de uma maneira padrão por qualquer agente JMX.

Um MBean é uma classe Java não abstrata que contém:

um construtor público;

a implementação da interface MBean correspondente ou a interface de um MBean dinâmico (*DynamicMBean*);

opcionalmente, a implementação da interface *NotificationBroadcaster*.

As classes que implementam a sua própria interface MBean são denominadas MBeans padrões. É o tipo mais simples de MBean disponível na especificação do JMX. As classes que implementam a interface *DynamicMBean* são denominadas Mbeans dinâmicos. Estas classes permitem que algumas das suas características e funcionalidades internas sejam controladas em tempo de execução.

Os agentes JMX permitem que se manipulem ambos os tipos de MBeans de forma transparente, ou seja, independente do tipo das aplicações de gerenciamento. Desta forma, o tipo de interface que o MBean implementa determina como ele será desenvolvido e não como ele será gerenciado.

Quando se desenvolve uma classe Java através de uma interface MBean padrão, os recursos a serem gerenciados são acessados diretamente através dos seus atributos e operações. Os atributos são entidades internas disponibilizadas através de métodos de recuperação (*get*) e de alteração (*set*). As operações são os outros métodos da classe que estão disponíveis para a aplicação de gerenciamento. Todos estes métodos são definidos estaticamente na interface do MBean e são visíveis (externalizados) aos agentes através da característica de introspeção do Java. Esta é a forma mais direta de se desenvolver o gerenciamento de um novo recurso.

A seguir está apresentado um exemplo da definição de uma interface em Java de um MBean denominado *MinhaClasse*:

```
public interface MinhaClasseMBean {
public Integer getEstado();
public void setEstado(Integer e);
public void reset();
}
```

A seguir está apresentado o código fonte da classe do MBean exemplo denominado *MinhaClasse* e que implementa a interface *MinhaClasseMBean*:

```
public class MinhaClasse implements MinhaClasseMBean {
private Integer estado = null;
private String oculto = null;
public Integer getEstado() {
return(estado);
}
public void setEstado(Integer e) {
estado = e;
}
public String getOculto() {
return(oculto);
}
public void setOculto(String e) {
```

No exemplo apresentado, os métodos *getOculto* e *setOculto* da classe do Mbean *MinhaClasse* não farão parte da interface de gerenciamento porque não aparecem

na especificação da sua interface.

Quando se desenvolve uma classe Java através de uma interface MBean dinâmico, os atributos e as operações são acessados indiretamente através de chamadas de métodos. Ao invés da introspeção, os agentes JMX devem chamar um método que encontre o nome e a natureza dos atributos e das operações do MBean. Quando o agente JMX for acessar um atributo ou operação, ele inicialmente chama um método genérico cujos argumentos contém o nome do método ou da operação. Através dos MBeans dinâmicos é possível que rapidamente se gerenciem recursos ou aplicações já existentes, desde que estes sigam a especificação do JMX.

Os MBeans dinâmicos são recursos gerenciados que são monitorados através de uma interface predefinida a qual externaliza os atributos e as operações em tempo de execução: a interface DynamicMBean. Ao invés de externalizá-los diretamente através dos nomes dos métodos, os MBeans dinâmicos implementam um método que retorna as assinaturas de todos os atributos e operações disponibilizadas elas suas interfaces.

Todos os métodos de um MBean devem ser implementados de forma que a classe do MBean possa ser instanciada (não pode ser uma classe abstrata) e que esta instância possa ser gerenciada por uma aplicação de gerenciamento.

Modelo de Notificação

O modelo de notificação do JMX permite que aplicações que necessitem receber notificações expontâneas se registrem em um MBean de broadcast, que faz parte do modelo de notificação do JMX. Este modelo é composto pelos seguintes componentes:

um tipo de evento genérico, denominado *Notification*, que representa qualquer tipo de notificação de gerência;

a interface *NotificationListener*, que precisa ser implementada pelos objetos que desejam receber notificações geradas por MBeans (consumidores de notificações);

a interface *NotificationFilter*, que precisa ser implementada pelos objetos que atuarão como filtros de notificações;

a interface *NotificationBroadcaster*, que precisa ser implementada pelos MBeans que estiverem monitorando recursos que possam gerar informações expontâneas (geradores de notificações).

O modelo de notificação do JMX está representado na Figura 2, a qual apresenta os passos envolvidos no processo de registro e de envio de uma notificação desde um MBean gerador até um MBean consumidor.

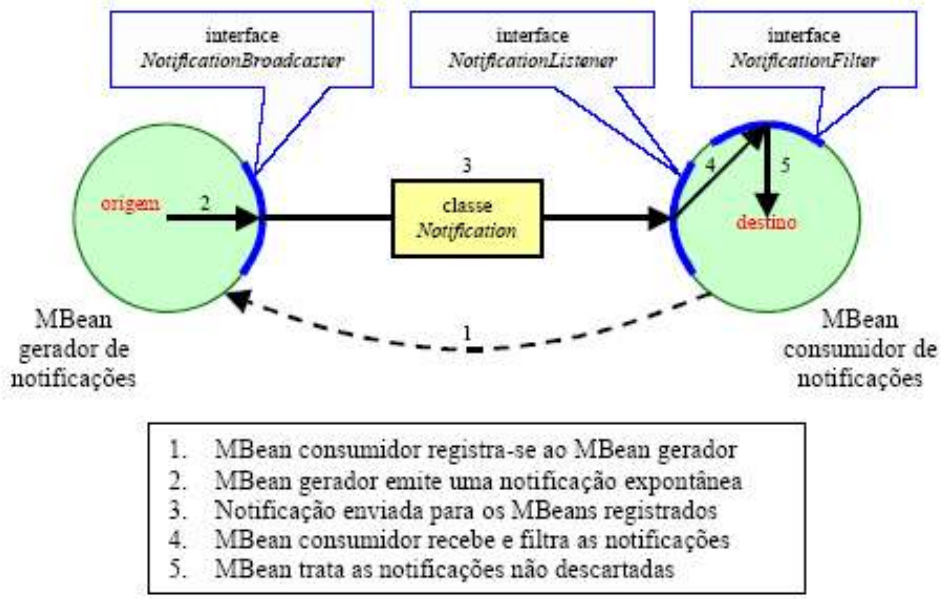


Figura 2: Modelo de notificação do JMX

Usando tipos de eventos genéricos, o modelo de notificação permite que qualquer consumidor de notificações receba todos os tipos de eventos de um gerador de notificações. O filtro de notificações é utilizado por um consumidor para especificar quais os tipos de notificações que ele precisa.

3.2.1.2 Nível Agente

O nível agente é uma especificação da implementação dos agentes JMX. Os agentes JMX são responsáveis pelo controle direto dos recursos e por tornarem estes recursos disponíveis para as aplicações de gerenciamento remotas. Este nível usa o nível instrumentação para definir os recursos e serviços que o agente JMX fornecerá ao sistema de gerenciamento. Um agente JMX consiste de um Mbean servidor, um conjunto de serviços básicos de gerência e pelo menos um adaptador de comunicação ou um conector JMX.

As aplicações de gerência acessam MBeans de um agente através de um adaptador de comunicação ou de um conector JMX e usam os seus serviços. Por outro lado, um agente JMX não precisa ter conhecimento da aplicação de gerência que o está gerenciando. A arquitetura do JMX permite que uma aplicação de gerência execute as seguintes operações sobre um agente JMX:

gerencie MBeans já disponíveis, recuperando ou alterando os valores dos seus atributos;

receba notificações emitidas por MBeans;

carregue novos MBeans para que possam ser instanciados e registrados;

instancie e registre novos MBeans já carregados.

Agentes JMX são implementados por desenvolvedores de sistemas de gerenciamento os quais podem construir os seus produtos de uma forma padrão sem que para isto tenham que entender a semântica dos recursos gerenciados ou as funcionalidades das aplicações de gerência envolvidas no sistema.

MBean Servidor

O MBean servidor é o núcleo do agente JMX e é responsável pelo registro dos MBeans no agente e por prover os serviços necessários à sua manipulação. Todas as operações de gerência executadas sobre os MBeans são feitas através das interfaces do MBean servidor (*MBeanServer*).

A Figura 3 representa como uma operação de gerenciamento é propagada desde uma aplicação de gerência até um MBean registrado no agente JMX. O exemplo ilustra a propagação de um método para recuperar o conteúdo do atributo *estado* de um MBean desde uma aplicação de gerência, tanto através de uma interface de invocação estática quanto através de uma interface de invocação dinâmica.

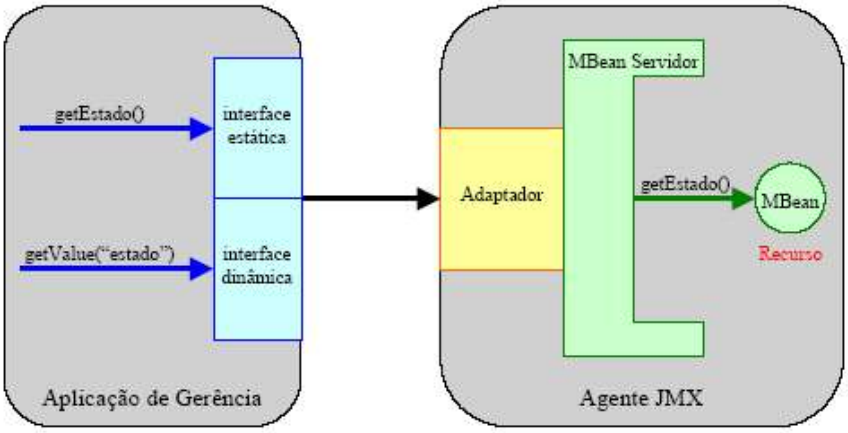


Figura 3: Propagação de uma operação sobre um Mbean

Os MBeans podem ser registrados em um MBean servidor tanto por aplicações de gerência quanto por outros MBeans. Os seguintes tipos de MBeans podem ser registrados:

MBeans que representem recursos gerenciados, do tipo aplicações, sistemas ou recursos de rede, e que tenham sido desenvolvidos de acordo com a especificação do JMX;

MBeans que acrescentem funcionalidades de gerência ao agente JMX;

MBeans de implementação dos adaptadores de protocolo ou dos conectores JMX.

Carregamento Dinâmico

O serviço de carregamento dinâmico é representado pelo serviço MLet (programa Java de gerenciamento para ser inserido em uma página da Internet) o qual é utilizado para instanciar MBeans obtidos de uma URL (Universal Resource Locator) remota.

Um serviço MLet permite que se instancie e registre em um MBean servidor um ou mais MBeans vindos através da rede. Isto é feito através do carregamento de um arquivo texto contento a especificação do MLet o qual contém as informações de cada um dos MBeans a serem carregados desde o local especificado pela URL.

A Figura 4 descreve a operação de carregamento, registro e instanciação de um MBean desde uma URL remota. Neste exemplo, a classe do MBean *obj1* está disponível na máquina local onde o agente JMX está sendo executado, enquanto que o MBean *obj2* foi carregado por um navegador Internet, fazendo o papel de uma aplicação de gerência, através de uma página HTML que contém a descrição e a fonte do MBean a ser instanciado no agente.

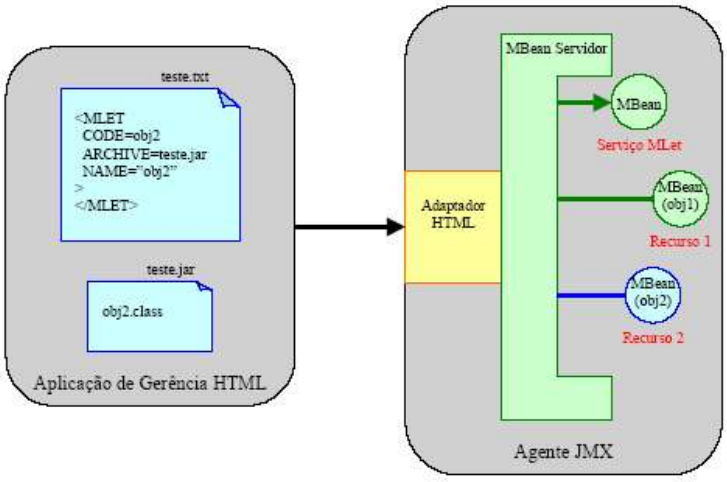


Figura 4: Operação do Serviço MLet

O serviço MLet é implementado como um MBean e registrado no MBean servidor de forma que ele pode ser utilizado tanto por aplicações de gerenciamento quanto por outros MBeans registrados no agente JMX.

3.2.1.3 Nível Gerente

O nível gerente corresponde a uma especificação da implementação dos gerentes JMX. Este nível define as interfaces de gerência e os componentes que podem operar sobre os agentes JMX ou, até mesmo, sobre hierarquias destes agentes.

A combinação do nível gerente com os níveis agente e instrumentação provê uma arquitetura completa para o projeto e o desenvolvimento de soluções de gerenciamento de sistemas. A tecnologia JMX traz várias facilidades para estas

soluções: portabilidade, desenvolvimento de funcionalidades de gerência sob demanda, serviços de gerenciamento dinâmicos e móveis, segurança.

Os componentes de um sistema de gerência JMX devem:

Prover uma interface para que aplicações de gerenciamento interajam com um agente JMX através de conectores JMX;

Distribuir informações de gerenciamento para agentes JMX de vários níveis;

Consolidar informações de gerenciamento provenientes de agentes JMX de vários níveis em perspectivas lógicas específicas para que se tornem relevantes para as operações e expectativas de um determinado usuário; Prover mecanismos que garantam a segurança do sistema de gerenciamento como um todo.

3.2.2 APIs de Protocolos de Gerência Adicionais

As APIs de integração com outras tecnologias de gerência são independentes deste modelo de três níveis do JMX (Figura 1). Algumas destas APIs de integração já estão identificadas pela Sun:

API de gerência SNMP;

API de gerência CIM/WBEM (Common Information Model/Web-Based Enterprise Management);

API de gerência CORBA;

API de gerência TMN.

Estas APIs de protocolos de gerência adicionais provêem uma especificação para permitir que agentes JMX possam interagir com ambientes de gerenciamento já disponíveis no mercado. Já estão completamente especificadas pela Sun as interfaces com sistemas de gerenciamento que se utilizam dos seguintes protocolos padrões de gerência:

SNMP [JMXSNMP 1999]: representa e manipula objetos SNMP como classes Java em agentes JMX para que estes possam ser gerenciados por aplicações de gerência SNMP;

CIM/WBEM [JMXWBEM 1999]: representa e manipula objetos CIM como classes Java em agentes JMX para que estes possam ser gerenciados por aplicações de gerência WBEM.

Desenvolvedores de aplicações de gerência podem utilizar estas APIs para interagir com ambientes de gerenciamento baseados nestes protocolos padrões, possivelmente encapsulando esta interação em um recurso gerenciável JMX. Estas APIs Java auxiliam desenvolvedores de sistemas de gerência a construir aplicações independentes de plataforma para os protocolos padrões de gerenciamento mais comuns da indústria. Desta forma, novas soluções de gerência podem integrar-se com infra-estruturas já existentes e sistemas de gerenciamento já existentes podem tirar vantagem de aplicações de gerenciamento baseadas na tecnologia Java.

3.2.3 Considerações Gerais

A especificação do JMX, que visa suportar o desenvolvimento de aplicações dinâmicas de gerência baseados na tecnologia Java e integradas à Internet, incorpora os recursos inerentes à linguagem de programação Java ao paradigma gerente/agente proposto pela arquitetura do TMN (Figura 5). A proposta desta arquitetura é viabilizada através da utilização dos componentes de software do Java: os JavaBeans [Core Java V2].

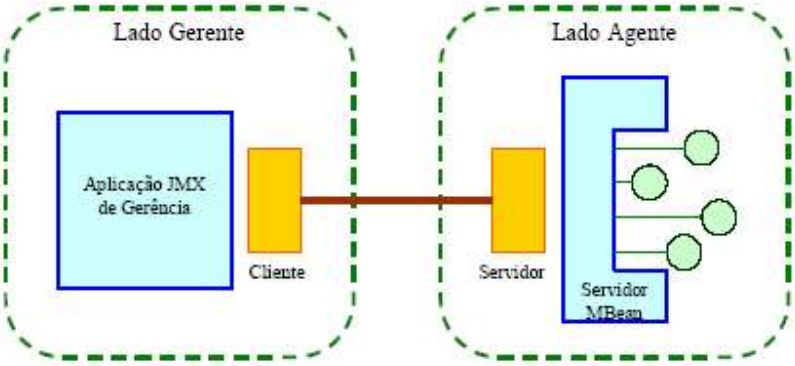


Figura 5: Arquitetura gerente/agente do JMX

A tecnologia JMX se baseia em um objeto gerenciável servidor que atua como uma agente de gerenciamento e que pode ser executado em qualquer dispositivo que esteja habilitado para a linguagem Java. O JMX especifica uma forma padrão de, através deste objeto gerenciável servidor, habilitar em qualquer dispositivo, serviço ou aplicação Java, a capacidade de tornar-se gerenciável.

Qualquer serviço de gerência JMX é um módulo independente que pode ser carregado no agente de acordo com a necessidade. Esta arquitetura de gerência baseada em componentes implica que soluções desenvolvidas de acordo com o JMX são escalonáveis de acordo com as necessidades de utilização do agente e dos requisitos de gerenciamento do sistema. Portanto, todos estes serviços podem ser carregados, descarregados ou atualizados dinamicamente pelo sistema.

Agentes JMX são capazes de serem gerenciados por navegadores HTML (Hypertext Markup Language) ou por protocolos de gerência tais como SNMP e WBEM. A especificação do JMX inclui a definição da API de um gerente SNMP e também de um cliente WBEM.

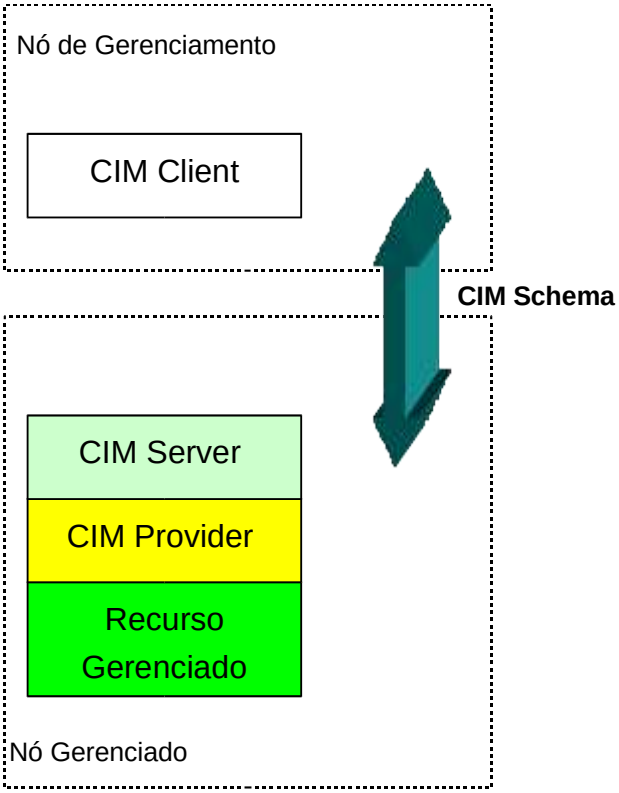
3.3 OpenPegasus

O Pegasus é uma implementação dos padrões DMTF CIM e WBEM. Ele foi desenvolvido para ser portável e altamente modular. Com codificação em C++ de forma a traduzir o conceito de objetos do CIM em um modelo de programação, mas mantendo a velocidade e eficiência de uma linguagem compilada. O Pegasus pode ser montado e roda na maioria das versões de UNIX, Linux e Windows.

O projeto Pegasus é open-source. Seu desenvolvimento está sob os auspícios do The Open Group, contudo, as contribuições para o projeto não está limitada aos membros do The Open Group

É uma plataforma para a construção de Aplicações de Gerenciamento e tem como foco implementações de grande volume de servidores. O objetivo não é o de gerenciar sistemas, mas sim de tornar os sistemas gerenciáveis através da padronização do ambiente de instrumentação. O gerenciamento dos sistemas é efetuado por aplicações de gerenciamento proprietárias.

3.3.1 Componentes da Arquitetura



CIM: Common Information Model – Define o schema utilizado para representar objetos reais que estão sendo gerenciados. Na implementação Pegasus os objetos CIM são representados internamente como classes C++.

CIM Client: envia requests de Operações CIM e recebe e processa as respostas das Operações CIM.

CIM Server: recebe e processa os requests de Operações CIM e emite as respostas para as Operações CIM.

CIM Provider: é responsável pelo processamento das Operações CIM em um ou mais recursos gerenciados. Ele faz o mapeamento entre a Interface CIM e a Interface específica do recurso.

Recurso Gerenciado: é a entidade que está sendo gerenciada (como memória, processo, sistema, aplicação, rede) mais o instrumentação específica o recurso que é capaz de monitorá-lo e controlá-lo.

Pegasus Server – Servidor WBEM/CIM com interfaces para providers e clients. O servidor trata das operações DMTF CIM, as indications DMTF CIM em conjunto com os providers.

Pegasus Repositories – o Pegasus possui uma interface de repositórios de classes e um repositório simples de classes baseado em arquivos. Ele também inclui um repositório de instâncias.

Biblioteca de Clientes Pegasus – Ferramentas para construção de clientes Pegasus baseada nas interfaces C++, utilizando os protocolos WBEM HTTP/XML ou fazendo interface direta com o Pegasus.

Clientes de Teste Pegasus – Testes simples que foram criados como parte do processo de desenvolvimento. Estes testes utilizam um Servidor Web com um conjunto de módulos CGI e HTML que permitem a entrada de operações Pegasus a partir do browser através de formulários e de receber as respostas como páginas web.

Pegasus Provider Managers – O conceito do Pegasus de Plugable Provider Managers permite múltiplos gerenciadores de provedores com interfaces diferentes.

Biblioteca C++ de Interfaces de Providers Pegasus – Interfaces para montar provedores Pegasus utilizando interfaces C++.

Biblioteca CMPI de Interfaces de Providers Pegasus – Interfaces para montar provedores Pegasus utilizando interfaces CMPI definidas em C.

Pegasus Providers – Para ilustrar a utilização dos serviços Pegasus incluindo providers para teste e demonstração, existem numerosos providers operacionais para tratar determindas classes DTMF como a classe CIM_ComputerSystem.

Pegasus Control Providers – Providers especiais que exigem acesso direto ao servidor para informações – eles são considerados providers internos Pegasus. Existem vários tipos destes providers definidos, incluindo informações de configuração e as classes DTMF interop.

Pegasus MOF Compiler – O compilador (cimmof) incluído é utilizado para instalar o MOF no Pegasus. Este compilador opera como um cliente Pegasus utilizando o Servidor CIM em execução para instalar as definições MOF.

3.3.2 Plataformas Suportadas

O Pegasus é testado em várias plataformas pelo grupo de desenvolvimento. Ele é geralmente suportado pelas seguintes plataformas e compiladores:

Plataforma	Compiladores
AIX	VisualAge C++ Version
HP-UX	HP aC++
Linux Itanium	gcc
Linux IA-32	gcc (versions 2.9x e 3.xx)
Windows 2000	Microsoft Visual C++ Ver 6 e Microsoft .Net compiler Version 7
Windows XP	Microsoft Visual C++ Ver. 6 e Microsoft .Net compiler Version 7

3.3.3 Instalação

O Pegasus está disponível através de Snapshots ou via CVS.

O CVS (Common Version System) é um sistema de controle de versão para desenvolvimento de software livre. Através do CVS se obtém o código fonte, os make files e documentação. O CVS já está presente nos sistemas Linux e pode ser baixado do site http://www.cvshome.org/ para instalação em sistemas Windows.

O snapshot é o congelamento do CVS em uma dado momento, quando são gerados os releases. A vantagem de baixar um snapshot é ter a certeza que ele é uma versão estável e testada. O ultimo snapshot disponível para download é o de versão 2.3.2. Aqueles que desejam compilações mais atuais devem obter o CVS, que incorpora todas atualizações mais recentes.

A obtenção dos códigos é fácil e não traz dificuldades. O snapshot da versão 2.3.2 possui 9.6MBytes em um único arquivo compactado. Este arquivo está disponível em ZIP para ambientes Windows e em GZ para ambientes Linux.

Dependências

A instalação do Pegasus exige a presença de alguns aplicativos instalados na máquina. Estes dependem da plataforma que está se instalando e são os que seguem:

1 – GNUMAKE – para simplificar a montagem do Pegasus em multi-plataformas.

2 – MU.EXE – Para minimizar as diferenças entre o Windows e Linux. Este aplicativo é utilizado apenas em ambiente Windows.

3 - FLEX e BISON – estas ferramentas são utilizadas para desenvolver o compilador MOF e o parser WQL.

4 – OpenSSL – se desejar utilizar o protocolo de comunicação SSL.

3.3.4 Compilação e Montagem

O pacote do Pegasus inclui os arquivos de batch para compilação e montagem dos programas. Os procedimentos também estão bem definidos na documentação anexa.

Pelo que pudemos observar o usuário que não faz parte do grupo de desenvolvedores deve utilizar um snapshot para obter o Pegasus. A obtenção através CVS é utilizado pelos desenvolvedores e esta sujeito a algumas inconsistências, o grupo de discussão possui em média 20 emails por dia relacionados à montagem do sistema através do CVS.

Inicialização do Repositório

Antes de utilizar o Pegasus deve-se popular o repositório. Existe um makefile que efetua esta tarefa, mas existem instruções de como fazer manualmente também.

Testes

O Pegasus inclui vários conjuntos desenvolvidos para testes que compreendem:

Clientes de Teste – vários clientes foram criados para testes: TestClient, Client, CLI, ipinfo, osinfo, WbemEsec, etc. Estes programas exigem que o Servidor completo com repositório esteja rodando

Prodivers de Teste – existem providers de teste para a maioria dos tipos de providers.

Suite de Testes Fim-a-Fim – conjunto de operações de testes que cobrem a maioria das operações CIM. Executa um conjunto extensivo de testes fixos e compara os resultados com valores pré-definidos.

3.3.5 Documentação

A documentação disponível é bastante detalhada e possui instruções específicas para instalação em sistemas operacionais Windows e Linux.

Existem instruções também de como criar e utilizar os certificados SSL.

A documentação falha quando trata do desenvolvimento de novos clientes e providers. Se os clientes e providers existentes não atenderem às necessidades será necessário procurar obter informações adicionais na literatura.

4. Comparações

Funcionalidades			
	NAGIOS	JMX/OpenNMS	OpenPegasus
Padrão	SNMP	WBEM	WBEM
Licença (free)	GPL	MIT	GPL

Portabilidade			
	NAGIOS	JMX/OpenNMS	OpenPegasus
Linux	sim	sim	sim
UNIX	não	sim	sim
Microsoft W2k	não	sim	sim

Suporte			
	NAGIOS	JMX/OpenNMS	OpenPegasus
Fabricantes	Ethan Galstad	SUN	HP/IBM/EMC[2]
Forum	Ethan Galstad	OpenWBEM	The Open Group
Código Fonte	sim	sim	sim
Binários	sim		não

Propósitos			
	NAGIOS	JMX/OpenNMS	OpenPegasus
Ger. Redes	sim	sim, através de proxy	sim, através de provider
Ger. Servidores	sim	sim	sim
Ger. Aplicações	sim	sim (java)	sim

Propósitos			
	NAGIOS	JMX/OpenNMS	OpenPegasus
Notificações (email /wap / sms)	sim	sim	n/a
Detecção de hosts “vivos”	sim	sim	n/a
hierarquia de hosts	sim	sim	n/a
Agendamento de checagens	sim	sim	n/a
Eventos pré-estabelecidos	sim	sim	n/a
checagens indiretas	sim	sim	n/a

5. Conclusões

•Estamos comparando “produtos” destinados a Gerenciamento de Servidores (JMX e Pegasus) com outro “produto” mais voltado a gerencia de ativos de rede, o NAGIOS.

•O Pegasus e JMX também podem gerenciar equipamentos de rede, mas para isto precisam de proxies para interface com o mundo SNMP

•As soluções analisadas são todas suportadas por comunidades ativas e engajadas, facilitando a troca de experiências e resolução de bugs.

•O Pegasus não é um produto de gerenciamento, seu propósito é servir como base para o desenvolvimento de soluções de gerenciamento.

•O NAGIOS é o mais indicado para que precisa de uma solução pronta para gerencia de redes para ser instalado em Linux.

•O JMX é uma solução da SUN destinado primordialmente para gerenciar aplicações JAVA.

•***As Soluções não são excludentes, e dependendo do propósito que se objetiva, as três podem ser implantadas no mesmo ambiente.***

6. Referências Bibliográficas

[Core Java V1] C. S. Horstmann, G. Cornell. Core Java 2, Volume 1: Fundamentals. Sun Microsystems, Inc. Prentice Hall. 1999.

[Core Java V2] C. S. Horstmann, G. Cornell. Core Java 2, Volume 2: Advanced Features. Sun Microsystems, Inc. Prentice Hall. 2000.

[JDMK 1999] Sun Microsystems, Inc. Java Dynamic Management Kit. Programming Guide. March 1999.

[JDMKWP 1998] Sun Microsystems, Inc. Java Dynamic Management Kit. A White Paper. February 1998.

[JMX 1999] Sun Microsystems, Inc. Java Management Extensions. June 1999.

[JMXWBEM 1999] Sun Microsystems, Inc. Java Management Extensions. CIM/WBEM APIs. August 1999.

[JMXSNMP 1999] Sun Microsystems, Inc. Java Management Extensions. SNMP Manager API. August 1999.