UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CIÊNCIA DA COMPUTAÇÃO

RODRIGO FRAGA MOHR

Análise de Ferramentas de Monitoração de Código Aberto

Monografia apresentada para obtenção do Grau
de Bacharel em Ciência da Computação pela
Universidade Federal do Rio Grande do Sul

Taisy Weber
Orientador

Porto Alegre, Dezembro de 2012

“Só é lutador quem sabe lutar consigo mesmo.“
— CARLOS DRUMMOND DE ANDRADE

AGRADECIMENTOS

Agradeço ao meu pai, mãe e irmão, por todo o apoio e carinho durante toda a minha vida e, em especial, nessa mais breve realização.

A todos os meus familiares que, uns mais perto, outros mais longe, todos mostraram confiança no meu trabalho.

A todos os meus amigos de dentro da faculdade, em especial aos do time QCB por todo os momentos felizes passados juntos. Iremos em busca de mais campeonatos!

A todos os meus colegas de trabalho da Dell pelo entendimento da importância desse momento. Em especial aos meus colegas Hulbe, Lilyan e Araray pela ajuda na elaboração do tema e conceitos.

À orientadora Taisy, por toda a ajuda, preocupação e atenção dedicados nesse último semestre.

Também agradeço à UFRGS, por todo o ensino de excelente qualidade disponibilizado durante esses últimos 5 anos.

Muito obrigado a todos. Vocês fizeram e fazem um papel muito importante na minha vida e foram essenciais para a conclusão deste trabalho.

SUMÁRIO

LISTA DE ABREVIATURAS E SIGLAS

CPU	Central Processing Unit
CSV	Comma-Separated Values
FTP	File Transfer Protocol
HTTP	HyperText Transfer Protocol
ICMP	Internet Control Message Protocol
IETF	Internet Engineering Task Force
IP	Internet Protocol
IT	Information Technology
JMX	Java Management Extensions
MSDN	MicroSoft Developer Network
NAGIOS	Nagios Ain't Gonna Insist On Sainthood
NRPE	Nagios Remote Plugin Executor
PDF	Portable Document Format
RAM	Random-Access Memory
SNMP	Simple Network Management Protocol
SSH	Secure Shell
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
URL	Uniform Resource Locator
WMI	Windows Management Instrumentation
XML	Extensible Markup Language

LISTA DE FIGURAS

LISTA DE TABELAS

RESUMO

O uso da Internet vem crescendo cada vez mais nos últimos anos. Estamos chegando em um nível que nos acostumamos com os erros cada vez mais comuns de serem vistos nos sites e das mais variadas empresas. Para lutar contra isso, é necessário o investimento em ferramentas de monitoração para detectar os problemas antes que o usuário perceba que existe algo errado.

O objetivo deste trabalho é avaliar e comparar três ferramentas de monitoração de código aberto: Zabbix, Nagios e Zenoss. O mercado de ferramentas livre também vem crescendo muito e esse é um mercado a ser explorado e analisado com cuidado. Ter em mãos softwares gratuitos que são capazes de disponibilizar todos os recursos necessários para efetivar uma monitoração precisa e simples é um ponto que poderá facilitar o desenvolvimento de qualquer empresa.

Serão avaliados aqui os mais variados pontos de cada uma das ferramentas escolhidas. Será feita uma análise desde o nível de funcionamento de rede, passando pela documentação existente, interface gráfica e configuração da ferramenta. O objetivo é ter certeza que todas elas são capazes de efetuar as tarefas básicas de uma monitoração com sucesso e, ainda, compará-las para indicar qual se sobressai em cada critério observado.

Também serão efetuados estudos de casos práticos onde cada uma das ferramentas será aplicada em um ambiente virtual para que sejam efetuadas medições de performance, facilidade de uso e capacidade das ferramentas.

De uma maneira geral, a ferramenta Zabbix mostrou ser a mais completa de todas, incluindo um bom poder computacional com uma interface bastante boa. O Nagios não tem muito foco na interface mas provou ser bastante adaptável ao facilitar o desenvolvimento de plugins. Pode ser observado que o Zenoss teve um grande investimento na sua interface gráfica, que pareceu ser o foco para essa ferramenta.

Palavras-chave: Monitoração, Ferramenta Livre, Servidor, Desempenho, Falha, Interface.

Analysis of Open Source Monitoring Tools

ABSTRACT

The use of Internet has been growing more and more in last years. We are reaching a level which we are getting used to each time more common errors are being seen in the websites of most varied companies. To fight against that, it is necessary to invest in monitoring tools to detect problems before the user realizes there is something wrong.

The goal of this work is to analyse and compare three open source monitoring tools: Zabbix, Nagios and Zenoss. The market of free tools is also growing a lot and this is a market to be explored e analysed carefully. To have in hands free softwares that are capable of make available all resources necessary to make an accurate and simple monitoring is a point that will make easier the development of every company.

The most varied aspects of the chosen tools will be validated here. It will be done an analysis of since the behaviour in the network side, going to the existing documentation, graphic interface and tool configuration. The goal is to make sure all of them are capable of, with success, provide basic tasks of a monitoring and, still, compare them to indicate which of them will be best in each observed criteria.

It will also be done case studies in which each tool will be applied to a virtual environment so that it will be possible to measure performance, ease of use and capacity of the tools.

In a global way, the tool Zabbix was the one that showed to be more complete, including a good computation power with a very nice interface. Nagios does not have a big focus on the interface but has proven to be very adaptable by making easier the development of plugins. It was also possible to observe that Zenoss had a great investment on its graphic interface, which seems to be the focus for this tool.

Keywords: Monitoring, Open Tool, Server, Performance, Fault, Interface.

1 INTRODUÇÃO

O constante crescimento da utilização da Internet em todo o mundo faz com que, cada vez mais, seja necessário ter uma atenção especial às aplicações e servidores das empresas. A monitoração faz um papel essencial para que os clientes tenham uma experiência livre de problemas acontecendo nas máquinas e aplicações da empresa.

O mercado de ferramentas de código aberto vem crescendo muito nos últimos anos (URLOCKER, 2009). Isso mostra o grande impulso que as ferramentas de monitoração livres estão tendo no mercado.

1.1 Motivação

O tema do trabalho foi escolhido pois segue a linha de atividades profissionais do autor. Trabalhar no setor de monitoração de uma grande empresa da área de IT é um grande incentivo.

Foram escolhidas 3 ferramentas de código aberto para serem analisadas: Zabbix, Nagios e Zenoss. Essas três ferramentas também estão sendo analisadas na empresa do autor e isso favoreceu sua escolha.

A opção por usar ferramentas de código livre é pela facilidade de mudança que elas apresentam. São ferramentas que, se houver um estudo sobre a melhor maneira de usar cada uma delas, podem desempenhar um grande papel na monitoração de uma empresa.

1.2 Objetivos

O principal objetivo é analisar alguns critérios que são considerados essenciais para uma ferramenta de monitoramento, de acordo com o autor. Facilitar a escolha entre essas três ferramentas uma vez que se conheçam as necessidades.

Também é um objetivo identificar oportunidades de melhorias nessas ferramentas. Como todas elas são de código aberto, isso poderia ser uma realidade plausível.

Ao final, analisar comparativamente as três ferramentas e chegar a conclusão de qual delas é a mais completa. Não será feita nenhuma análise voltada a alguma aplicação para manter a investigação o mais genérica possível.

1.3 Resultados Alcançados

A ferramenta que teve a melhor performance geral e se mostrou a mais completa foi o Zabbix. O Nagios mostrou ser bastante completo mas com pouco investimento na parte gráfica. Já o Zenoss pareceu ter muito investimento na parte gráfica, o que acabou fazendo

ela ficar meio confusa.

1.4 Organização do Trabalho

No Capítulo 2, será feita uma abordagem com a finalidade de esclarecer os conceitos utilizados neste trabalho. O objetivo é que o entendimento do autor sobre monitoração, ferramentas de monitoria e os protocolos utilizados esteja bem explicado.

Já no Capítulo 3 haverá um estudo sobre o funcionamento das três ferramentas escolhidas: Zabbix, Nagios e Zenoss. O estudo foi divido em três tópicos principais e igualmente importantes: como acontece a coleta de informação, como os problemas são detectados e de que forma as ferramentas auxiliam os usuários na análises dos problemas.

Os Capítulos 4 e 5 trazem uma análise mais prática sobre essas ferramentas. No primeiro será feito um estudo sobre uma série de critérios e atributos considerados importantes para uma ferramenta de monitoria. No segundo, haverá uma aplicação prática das ferramentas e observação dos resultados.

Para finalizar, o Capítulo 6 traz um encerramento do trabalho. Também serão abordadas aqui futuras oportunidades de trabalho e aprofundamento deste estudo.

2 CONCEITOS BÁSICOS

Antes de começar a descrever o funcionamento das ferramentas de monitoração escolhidas e os motivos de escolha tomados, será formada uma base conceitual para facilitar o entendimento deste trabalho. O objetivo aqui é melhorar o entendimento de todos os conceitos para que o autor possa se expressar com mais precisão.

Serão revistos vários tópicos, com uma abrangência maior do que apenas monitoria. Haverá uma introdução sobre servidores conectados em rede (como eles se comportam, quais seus principais indicadores de performance sem especificar uma aplicação), ferramentas de monitoria (o que se espera delas, como elas são comumente usadas) e alguns protocolos de comunicação e monitoração (SNMP, SSH, WMI, entre outros).

2.1 Servidor

Um servidor de rede é um computador designado para processar requisições e entregar dados para outro (chamado cliente) computador em uma rede local ou na internet. Servidores normalmente são configurados com capacidades adicionais de processamento, memória e disco para serem capazes de lidar com a carga gerada pelos clientes. Tipos comuns de servidores em rede incluem:

- Serviços de Internet (lojas virtuais, redes sociais, entretenimento digital, ...);
- Proxy (Portão que faz requisições de internet no lugar de outro);
- FTP (transferência de arquivos);
- Jogos online;
- Bancos de Dados.

Na maioria dos casos, um servidor é um computador físico, apesar do crescimento da utilização de máquinas virtuais (servidor virtualmente alocado em outro, consumindo seus recursos). Em grandes empresas, existem muitos servidores co-existindo, o que faz com que seja necessário utilizar técnicas para evitar problemas físicos (dissipação de calor, sujeira e outros).

2.2 Monitoração

Monitoração é a observação e gravação regular de atividades acontecendo em um servidor remoto (em rede) ou local (físico) (COLLECTIVE, 2012). É um processo de constante coleta de informações em muitos aspectos diferentes. Monitorar é checar o

progresso/desempenho de uma determinada atividade/comportamento. É uma observação sistemática e muito importante.

Monitoria também envolve dar constantes retornos e comunicados do estado do objeto sendo monitorado para o administrador ou usuário (PHAAL, 1994). Habilidade de gerar relatórios é um aspecto importante na tomada de decisões para melhor o desempenho de um determinado servidor.

Entre os principais propósitos da monitoria estão:

- Analisar informações;
- Determinar que os recursos estão sendo utilizados da melhor maneira;
- Identificar problemas nas aplicações;
- Garantir que todas as atividades são executadas de forma correta;
- Facilitar a tomada de decisões ao resolver problemas.

Para uma boa monitoração, é essencial que exista uma ferramenta de monitoração com uma boa capacidade de processamento de informações. Uma habilidade importante é a capacidade de se gerar alertas, a partir da definição de limiares (chamados thresholds) (BALIS et al., 2002).

Em algumas empresas, existem times de profissionais dedicados exclusivamente à monitoração. As pessoas que trabalham nesse time, na área de suporte das empresas, são as responsáveis por resolver os problemas que acontecem nos servidores e que foram detectados pela monitoração.

Para que times assim consigam trabalhar com eficiência, é de extrema importância que a ferramenta possua um dashboard. Dashboard é um painel que mostra o estado de todos os monitores configurados.

2.2.1 Ferramentas de Monitoração

Uma ferramenta de monitoração é usada para executar checagens regulares nas aplicações da empresa com a finalidade de detectar problemas antes que o usuário da aplicação perceba (RAKOSHITZ et al., 2003). No mundo ideal da monitoria, nenhum usuário receberia um erro do tipo HTTP 503 (Serviço Indisponível).

Usuários de ferramentas de monitoria normalmente trabalham nas áreas de suporte das empresas, caso ela possua um setor específico para suporte. Dessa maneira, elas têm acesso direto ao todos os tipos de problemas que acontecem nas aplicações dessa empresa e estão acostumadas, e treinadas, para saber resolver os problemas.

Desse modo, a melhor maneira que um software de monitoramento pode ajudar o trabalho dessa pessoas é ajudando a detectar o mais rápido possível os problemas. Facilidade de uso, rapidez de alerta e visualização simplificada dos problemas são características desejáveis (NETO, 2001).

Também é importante a possibilidade e agilidade de validar o estado atual de cada monitor, uma vez que o usuário precisa ter certeza que o problema que estava investigando realmente parou de acontecer.

2.2.2 Limiar de Alerta

Sem a capacidade de gerar alertas, ferramentas de monitoria perdem grande parte de sua utilidade. Sem isso, seria necessário que hajam pessoas totalmente dedicadas à

utilização das ferramentas, tendo que checar manualmente todos os monitores. Isso seria extremamente custoso.

Um dos pontos críticos na hora de se definir um alerta é a escolha de um valor limiar (threshold). Ele será o valor comparado com uma expressão, calculada periodicamente para verificar o estado do monitor. Existem também monitores que não são apenas numéricos. Um monitor pode ser usado para checar se uma determinada URL abriu corretamente ao procurar na página por uma frase ou palavra. Como o foco deste trabalho está na monitoria dos servidores, esse tipo de monitoria será deixado de lado.

É crucial que a escolha do valor limiar de alerta de um monitor seja bem feita. Sendo feita uma escolha errada, o monitor poderá concluir que existe um problema quando não existe (situações de falso-positivo). A situação contrária também deve ser cuidada. Não se pode relaxar muito no valor e chegar ao ponto em que haja um problema mas o monitor determina que está tudo bem (falso-negativo).

Uma utilidade interessante de uma ferramenta de monitoração é a possibilidade de possuir um threshold dinâmico. Um exemplo seria, ao invés de ter um número fixo, calcular esse número a cada checagem para ver se o monitor deve alertar. Isso facilitaria a adaptabilidade do monitor a mudanças permanentes no ambiente monitorado (um exemplo seria uma expansão do site, o que aumentaria o tráfico nas máquinas). Infelizmente, nenhuma das ferramentas aqui estudadas permite a utilização de valores limiares dinâmicos nativamente.

2.2.3 Dashboard

Quando não for possível a geração de alertas, é imprescindível a presença de um dashboard. Isso seria um painel que facilitaria a visualização dos monitores e o estado de cada um deles. Mesmo havendo a geração de alertas, é muito importante que haja um lugar em que o usuário das ferramentas (pessoa que recebe os alertas) possa verificar se o problema continua acontecendo.

É muito importante que haja uma clara separação entre quais monitores estão com problema e quais estão indicando um funcionamento correto do sistema. Isso facilitaria as decisões das pessoas responsáveis por cuidar das aplicações sendo monitoradas.

Ainda existem os casos em que a geração de alertas não ocorre propriamente. Nesses casos, as pessoas responsáveis por cuidar dos alertas devem ficar atentas nos painéis para garantir que não existe nenhum monitor alertando sem que tenha alguém trabalhando nele.

2.3 Protocolos

Aqui será vista a maneira como cada um dos protocolos (SNMP, SSH, Telnet, JMX e WMI) funciona a nível de rede. Qual tipo de informação eles permitem verificar ou acessar também será um ponto importante a ser abordado.

2.3.1 SNMP

O SNMP (Simple Network Management Protocol) é um protocolo para gerência de redes. Ele é usado para coletar informações, e até mesmo ajustar configurações, de dispositivos ligados em rede, como servidores, impressoras, roteadores em uma rede utilizando o protocolo IP. Ele é um dos padrões de operação e manutenção de protocolos para a internet (FEIT, 1993). SNMP tem sido uma tecnologia essencial para o crescimento da Internet.

Esse protocolo fica localizado no nível de aplicação, utilizando UDP como para rea-

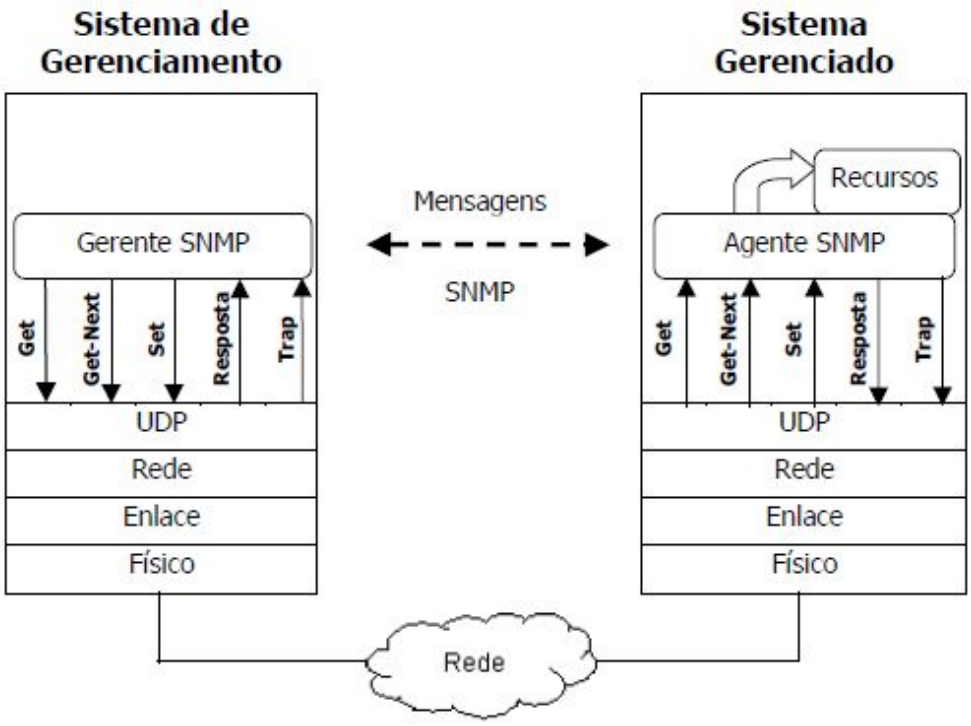


Figura 2.1: Modelo de Funcionamento do SNMP. Extraído de (KOCH, 2008)

lizar o transporte de informações (CASE et al., 1990). O funcionamento do SNMP exige que exista um gerente de SNMP que será responsável pela organização da rede e centralização da informação. Além dele, também deverá existir um agente SNMP instalado na máquina alvo a ser monitorada.

O protocolo funciona a partir da utilização de três operações genéricas, que leem ou alteram o conteúdo de objetos das máquinas alvos (KOCH, 2008). As três operações básicas são:

- Get: Essa é a principal operação utilizada por ferramentas de monitoria. Ela permite que o gerente consulte informações dos agentes.
- Set: Essa operação permite que o gerente modifique algum objeto do agente. Nas ferramentas de monitoria, pode ser usada na inicialização e configuração dos atributos necessários de monitorar.
- Trap: Também bastante usada na monitoração, essa operação permite que o agente notifique o gerente, sem necessitar de uma requisição, de algum evento.

Através do envio de mensagens SNMP, ilustradas na Figura 2.1, o servidor explicita qual objeto ele gostaria de visualizar (operação Get) e, na resposta, ele consegue visualizar o estado daquele objeto. Para um entendimento melhor, é possível imaginar como o Disco Rígido sendo um objeto e a requisição perguntaria qual o percentual de memória livre em determinada partição.

2.3.2 SSH

SSH (Secure Shell) é um protocolo de rede criptografado para comunicação de dados, serviços Shell remotos, execução de comandos de maneira segura entre dois computadores conectados em rede (YLONEN; LONVICK, 2006). Do mesmo modo que o protocolo

SNMP, esse protocolo também exige a instalação de um servidor SSH, na máquina monitorando, e um agente SSH, na máquina a ser monitorada.

Esse protocolo utiliza o método criptográfico de chave pública para autenticar computadores remotos. Como o objetivo desse protocolo é a comunicação protegida, é necessária a utilização do transporte de dados utilizando TCP.

Basicamente, toda a informação enviada é criptografada. Dessa maneira, são diminuídas as possibilidades de informações importantes (como usuário e senha) serem extraviadas no meio da comunicação. Uma vez que a conexão segura esteja estabelecida, o servidor gerente pode requisitar qualquer tipo de informação sobre o agente. Isso faz com que seja possível cobrir uma grande quantidade de monitores diferentes.

2.3.3 TELNET

Telnet é um protocolo de rede muito parecido com o SSH. Ele possibilita uma comunicação orientada a texto bi-direcional utilizando um terminal virtual de comunicação. A comunicação estabelecida através do Telnet não é segura, diferentemente do SSH, pois a informação não é criptografada (POSTEL; REYNOLDS, 1983). Ele é um protocolo orientado a conexão, utilizando o TCP.

Diferente dos protocolos vistos anteriormente, o Telnet não exige que haja um esquema servidor-agente configurado. A comunicação é toda feita por uma interface de linha de comandos. Nessa linha de comandos é possível obter informações sobre a máquina alvo a ser monitorada.

2.3.4 JMX

JMX (Java Management Extensions) é, diferentemente do visto até aqui, uma tecnologia de desenvolvimento de aplicações Java. Com a imensa utilização de Java no mundo todo, é imprescindível a capacidade de monitorar esse tipo de aplicação. E o Java acaba facilitando a monitoria pelo JMX.

Utilizando o console JMX é possível extrair informações importantes do funcionamento e execução da aplicação rodando na máquina alvo. Um exemplo é a obtenção da informação de quantidade de memória livre na pilha do Java. Essa é uma informação possível de ser extraída de um console JMX.

Em outras palavras, JMX não é um protocolo de comunicação nem de monitoria, e sim uma plataforma de execução e desenvolvimento que facilita a monitoria ao disponibilizar informações importantes de operação para usuários remotos.

2.3.5 WMI

WMI (Windows Management Instrumentation) é a infraestrutura para gerenciamento de dados e operações em sistemas operacionais do tipo Microsoft Windows (MSDN, 2012). Todo computador com sistema operacional Windows instalado possui uma instância do WMI rodando. Ele possibilita que usuários remotos possam obter informações sobre os servidores sem que seja necessário logar no computador.

O WMI funciona por um esquema de classes e entidades. Cada classe pode conter diversas entidades, que são os objetos a serem buscados para se obter informações mais detalhadas sobre a máquina alvo. A Figura 2.2 mostra um exemplo de uma classe e seus objetos disponíveis de serem consultados.

Isso possibilita uma grande quantidade de monitores de serem configurados, realizando checagens periódicas dos serviços. Entretanto, caso seja necessário monitorar al-

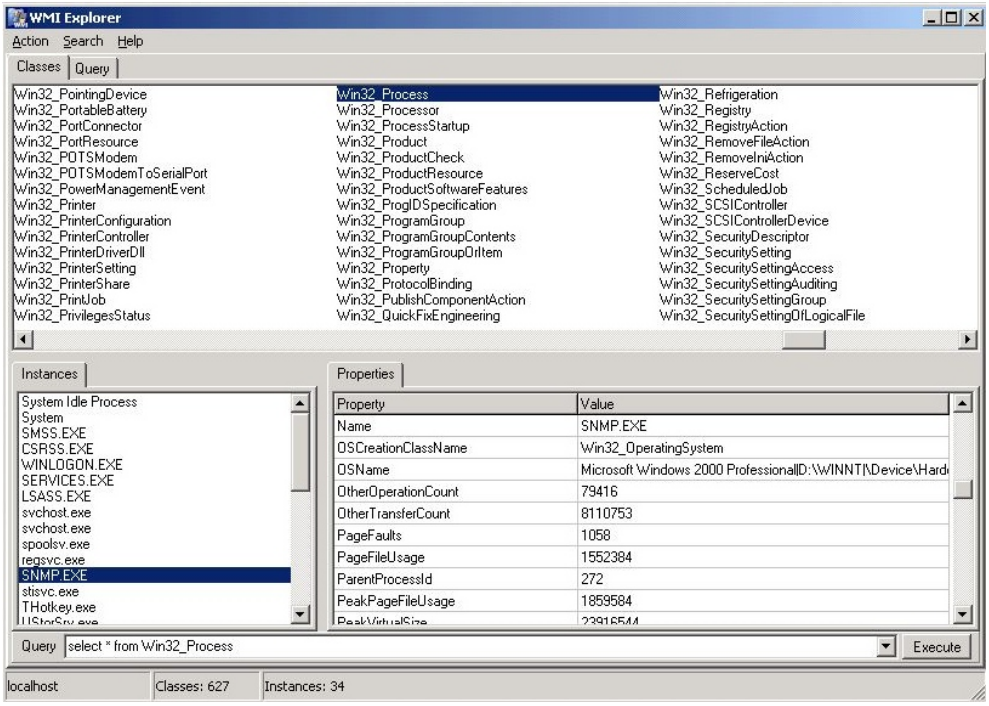


Figura 2.2: Modelo de Funcionamento do WMI

gum atributo que não possua uma classe configurada pela Microsoft, isso virá a ser um problema.

Agora que todos os principais conceitos foram devidamente apresentados, será feito um estudo mais profundo sobre cada uma das ferramentas escolhidas.

3 APRESENTAÇÃO DAS FERRAMENTAS

Nesse capítulo, será feito um estudo sobre as ferramentas. Na introdução de cada ferramenta será apresentado o seu histórico, com informações de quando ela foi criada, o que motivou a sua criação e quais as principais empresas utilizando essa ferramenta.

Após uma apresentação inicial das ferramentas, será feita uma análise mais profunda de como essas ferramentas trabalham. Será detalhado como elas coletam os dados (quais protocolos de comunicação são usados), como a detecção de erros aparece para o usuário e como essas ferramentas auxiliam para a investigação dos erros após a sua detecção.

3.1 Critérios de Escolha

O mercado das ferramentas de código aberto está crescendo significativamente nos últimos anos. Não só no lado de monitoria, mas em todas as áreas do mercado tecnológico (DUBIE, 2009).

Com esse pensamento em mente, foram pesquisadas as três principais ferramentas de monitoramento com código aberto. Outro critério de escolha foi a procura por ferramentas com propósitos levemente diferentes.

O Nagios foi escolhido por ser uma ferramenta voltada a usuários mais avançados, por usar muito pouco a interface gráfica e facilitar o desenvolvimento de plugins. Já o Zenoss é altamente voltado para a interface gráfica e compatibilidade com plugins já existentes. O Zabbix é uma mescla das outras duas, com potencial para ser a mais completa entre as três.

3.2 Zabbix

O projeto do Zabbix começou a ser desenvolvido em 1998, liderado por Alexei Vladishev. Três anos após o início do projeto, foi feita a primeira publicação da ferramenta já com o código aberto ao público. Apenas em 2005, foi criada a empresa ZABBIX SIA para fornecer serviços de suporte técnico aos seus clientes e usuários (ZABBIX, 2012).

O Zabbix possui diversos clientes de todo o mundo. Entre os principais clientes, existem alguns brasileiros:

- Dataprev;
- Lojas Renner SA;
- Petrobras;
- Procergs;

Figura 3.1: Zabbix: Locais de uso. Extraído de (ZABBIX, 2012)

- Serpro.

A missão dos profissionais do Zabbix é desenvolver uma solução de monitoração superior disponível e acessível para todos. Um dado interessante de ser observado é a Figura 3.1. Ela mostra todos os locais em que existe uma empresa utilizando o Zabbix. Em vermelho, a principal sede da empresa: Riga, Latvia.

3.2.1 Coleta de Dados

O Zabbix é um software que utiliza o tipo servidor-agente de funcionamento, inclusive possibilitando que mais de um servidor esteja rodando ao mesmo tempo, o que possibilitaria uma melhor performance e maior consistência de dados (OLUPS, 2010).

O sistema servidor conversa com o sistema agente (instalado em cada uma das máquinas que devem ser monitoradas) para requisitar informações sobre a máquina alvo. Essas informações são armazenadas em bancos de dados relacionais, que pode ser qualquer um dos abaixo:

- MySQL;
- PostreSQL;
- Oracle.

Se não houver a disponibilidade de um agente instalado na máquina a ser monitorada, essa monitoração também pode ser feita utilizando os seguintes protocolos (explicados em detalhes no capítulo anterior):

- SNMP;
- SSH;

- TELNET;
- JMX.

Com isso, essa ferramenta se mostra bastante completa ao cobrir uma grande área de possibilidades de monitoração. Todos esses tipos de monitoria podem co-existir. Isto quer dizer que podem haver monitores fazendo checagens pelos agentes instalados e, ao mesmo tempo, outros monitores estão utilizando o protocolo SNMP de comunicação para efetuar outras checagens.

3.2.2 Detecção de Erros

A detecção de erros e problemas acontece no servidor de processamento. As regras de gerenciamento das informações, como os valores limites de alerta, são armazenadas nos bancos relacionais que ficam nas máquinas servidores.

O Zabbix permite o monitoramento online onde é possível visualizar todos os dispositivos e seus principais indicadores de problema. Monitores de performance, segurança e utilização de CPU e memória são facilmente acessados através da interface web da ferramenta.

Um problema é caracterizado por uma expressão de monitoramento, configurada através da interface gráfica, ultrapassar seu valor limite. Um valor limite pode ser definido de muitas maneiras diferentes.

Existem vários tipos de valores que podem ser calculados para se comparar com um valor limite. Isso mostra um grande potencial existente nessa ferramenta, pois facilita muito a configuração de monitores que melhor se ajustem às necessidades dos usuários.

Um erro será detectado sempre que um valor calculado, "X", fizer a expressão "Y", calculada a partir de "N", retornar um valor falso. O valor limite definido estaticamente quando da criação do monitor é a variável "N". Para a expressão "Y", sua construção pode ser de uma das seguintes maneiras:

- X < N (X menor que N);
- X > N (X maior que N);
- X = N (X igual a N);
- X <> N (X diferente de N).

Quanto aos valores de "X", existem muitas maneiras de isso ser calculado. Ainda é possível usar a variável "T", um valor temporal a ser definido na criação do monitor. As principais, e mais comumente usadas, são:

- Diferença entre os dois últimos valores da expressão;
- Média de valores dos últimos T minutos;
- Último valor calculado;
- Maior/menor valor nos últimos T minutos;
- Soma dos valores nos últimos T minutos;
- Dia da semana/mês.

A expressão a ser calculada é o valor do monitor em si. Um exemplo para esclarecer esse atributo é pensar no uso de CPU. Em um monitor que cuidaria disso, esse valor seria o percentual do processador sendo utilizado.

Após a detecção de um problema pela ferramenta, o usuário pode ser alertado de diversas maneiras disponíveis, que vão desde um alerta por e-mail até executar um script no agente.

3.2.3 Análise de Problemas

A ferramenta fornece diversas soluções que auxiliam na investigação do usuário. Todos os monitores configurados podem ser observados de forma gráfica. Isso quer dizer que podem ser observadas tendências de comportamento.

Um exemplo para utilização disso seria o uso em um monitor de espaço em disco. Ao receber um alerta desse tipo, o usuário poderia entrar na interface da ferramenta e verificar o estado do disco nos últimos dias. Se a diminuição do espaço livre aconteceu em um espaço curto de tempo, pode ser que seja um único arquivo grande que foi criado no sistema. Se o espaço foi diminuindo gradualmente, pode ser que sejam arquivos de log do sistema.

Por possuir uma interface centralizada, com todas as informações, é possível acessar mais de um dado ao mesmo tempo e comparar seus resultados. O software disponibiliza diversas informações sobre o servidor que são atualizados em tempo real, e que podem vir a ajudar na investigação de algum problema.

3.3 Nagios

Ethan Galstad é o idealizar é criador da ferramenta Nagios. Em 1999, Ethan e seu time divulgaram a primeira versão, com o nome de NetSaint. Em 2002, devido a mudanças de estratégia, o nome foi mudado para NAGIOS, que na verdade é um acrônimo recursivo para "Nagios Ain't Gonna Insist On Sainthood"(Nagios não vai insistir em santidades), onde há uma referência para o antigo nome, "Sainthood"(NAGIOS, 2012).

Os produtos da empresa Nagios Enterprises são usados em todo o mundo e possuem diversos clientes que muito famosos nas suas áreas de atuação:

- AT&T;
- McAfee;
- Philips;
- Siemens;
- Sony;
- Toshiba;
- Ubisoft;
- Universal;
- Yahoo.

3.3.1 Coleta de Dados

A ferramenta Nagios não é uma ferramenta feita exclusivamente para monitoração de recursos de servidores. Ela possui diversos modos de funcionamento. O escopo deste trabalho é focar na monitoração de servidores.

Assim como o Zabbix, é possível a instalação de um agente do Nagios na máquina alvo. Dessa maneira, a comunicação toda é feita entre servidor e agente. O Agente fica na máquina a ser monitorada esperando requisições feitas pelo servidor. Ao receber uma requisição, por exemplo quantidade de espaço livre em um determinado disco, ele descobre a informação requisitada e retorna ao servidor.

A coleta de dados também pode ser feita utilizando protocolos como o SNMP e o SSH. Nessas opções, não é necessária a instalação de um agente do Nagios nas máquinas a serem monitoradas.

Uma outra opção disponível, que é uma opção bastante utilizada, é a instalação do plugin NRPE (Nagios Remote Plugin Executor) (PERVILä, 2007). Ele é instalado tanto na máquina a ser monitorada, quando na máquina monitorando. As checagens do Nagios são todas feitas localmente. Por isso, o NRPE se encarrega de realizar a comunicação com o servidor remoto. A máquina servidor imagina estar rodando a requisição internamente, pois apenas chama o plugin do NRPE, passando como parâmetro o plugin a ser executado e os parâmetros adicionais necessários para esse plugin.

O NRPE é um plugin padrão do Nagios, que faz a comunicação entre servidor e cliente. O seu objetivo é que o servidor acredite estar rodando todos os comandos localmente, através da sua versão do NRPE instalada. Toda a comunicação com o cliente fica, dessa maneira, escondida do Nagios (IMAMAGIC; DOBRENIC, 2007).

O Nagios é uma ferramenta com uma grande capacidade de modificação. Dessa maneira, qualquer tipo de comunicação estabelecida utilizando plugins pode ser efetuada com sucesso, independente do tipo de comunicação em vigor.

3.3.2 Detecção de Erros

Na execução dos plugins do Nagios é onde acontece toda a manipulação de resultados. Na chamada de um plugin, são passados os valores limítrofes daquele monitor (chamado no Nagios de serviço). Assim, o próprio plugin processa o resultado e explicita se aquele é um resultado bom, alarmante ou ruim.

Na Figura 3.2, pode ser observada a chamada do plugin que faz a checagem se uma máquina pode ser alcançada através da rede. Parâmetros importantes de serem observados são o -w"e o -c"que indicam os valores limites para indicar se um monitor está em estado alarmante (warning) ou crítico (critical).

É bom ressaltar que a definição de quais são os valores limites é feita na criação do monitor, pelo administrador do sistema.

Assim, resta apenas ao Nagios repassar ao usuário o estado notificado pelo plugin. Isso faz com que o Nagios tenha um menor controle sobre os resultados. Pelo outro lado, possibilita aos desenvolvedores de plugins controlarem muito bem os seus próprios resultados.

3.3.3 Análise de Problemas

A interface do Nagios não facilita muito a tarefa do usuário quando se deseja analisar comportamentos e tendências. A capacidade gráfica da ferramenta não foi um ponto investido quando do seu desenvolvimento.

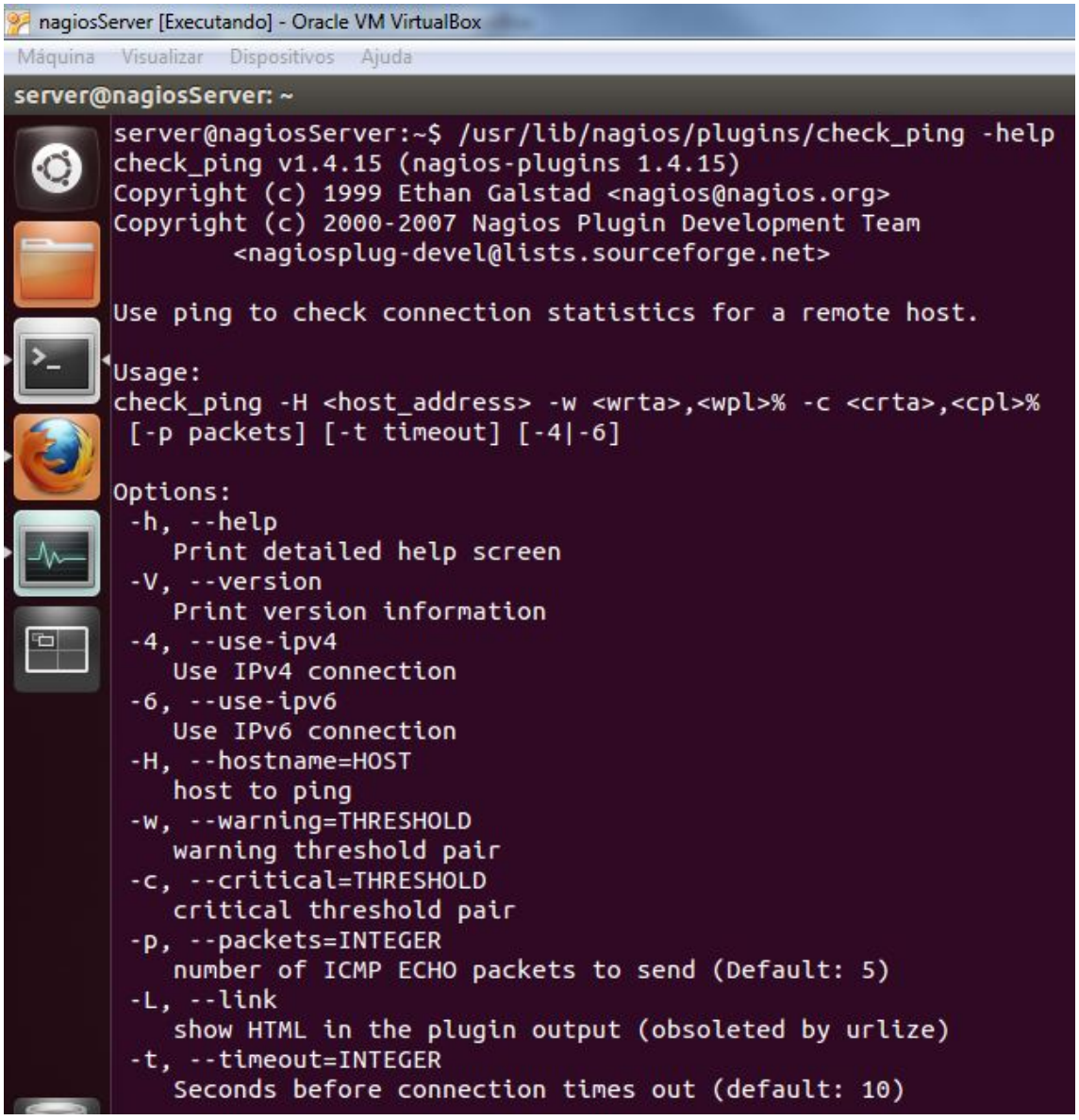


Figura 3.2: Nagios: Chamada do plugin de checagem de Ping

Por outro lado, é possível a instalação de plugins que facilitam a visualização gráfica dos eventos. Isso requer um pouco mais de investigação pelo lado do administrador da ferramenta, mas facilitaria para os usuários.

É possível extrair os dados, mais uma vez, utilizando plugins adicionais, e manipulá-los da maneira necessária. Um ponto a ser enfatizado aqui é que, com a instalação padrão do Nagios, não existem muitas possibilidades de análise dos dados. Entretanto, estendendo-se a ferramenta com a instalação de plugins é possível tornar a ferramenta mais completa.

3.4 Zenoss

Zenoss é a mais nova das três ferramentas aqui estudadas. Seu desenvolvimento começou em 2002. Em Agosto de 2005 foi fundada a empresa Zenoss Inc, uma parceria entre Erik Dahl (principal desenvolvedor) e Bill Karpovich (ZENOSS, 2012).

Entre seus principais clientes, encontram-se algumas grandes empresas como:

- Linkedin;
- Motorola;
- Exército dos EUA;
- Força Aérea dos EUA;
- Universidade de Chicago;
- VMWare.

3.4.1 Coleta de Dados

Toda a coleta de informação do Zenoss é feita utilizando os protocolo SNMP, principalmente, e SSH. Na máquina a ser monitorada, deve ser instalado um agente desse protocolo para que ele possa se comunicar com o servidor.

As informações estão sempre no cliente SNMP, basta o servidor buscar essas informações. Dessa maneira, a checagem é feita a partir da iniciativa do servidor que está monitorando, ao requisitar informações para o servidor monitorado.

Com isso, perde-se um pouco de poder de adaptabilidade, uma vez que as informações disponíveis são as que existem no cliente. Se, para um monitor novo, deseja-se extrair uma informação que não existia previamente configurada, talvez isso não seja possível.

Isso também pode causar um problema de escalabilidade, uma vez que todos os servidores agentes devem ser reconfigurados para possibilitar a visualização de um tipo diferenciado de monitoria.

No caso de a monitoração estar voltada para uma máquina cujo sistema operacional instalado é da família Microsoft Windows, existe a possibilidade de utilizar o serviço de WMI para coletar as informações necessárias.

Pelo lado do servidor, todo o gerenciamento de informações acontece em alguns chamados objetos de modelação do Zenoss. A Figura 3.3 ilustra todos os principais objetos utilizados para realizar a modelagem de um dispositivo.

Service	Group	Description	Runs As
ZenDisc	Modeling	Discovers new devices on the network	Linux daemon
ZenModeler	Modeling	Retrieves configuration detail and map resources to the classification model	Linux daemon
ZenWinModeler	Modeling	Discovers Windows-based services	Windows service
ZenStatus	Availability	TCP service monitoring (Telnet, SSH)	Linux daemon
ZenPing	Availability	Ping Availablity Monitoring (ICMP)	Linux daemon
ZenSNMP	Availablity	SNMP Availability Monitoring (SNMP)	Linux daemon
ZenWin	Availability	Windows service monitoring	Windows Service
ZenPerf	Performance	Performance collection (SNMP, Telnet, SSH)	Linux daemon
ZenSyslog	Event Collection	Syslog Event Collection	Linux daemon
ZenEventlog	Event Collection	Collecticon of windows events (WMI)	Windows Service
ZenTrap	Event Collection	SNMP trap collection (SNMP)	Linux daemon
ZenActions	Automated Response	Alerting (SMTP, SNPP) and command execution (SSH)	Linux daemon

Figura 3.3: Zenoss: Descrição dos objetos modeladores. Extraído de (LINUX WORLD MAGAZINE, 2006)

3.4.2 Detecção de Erros

O Zenoss é todo ele estruturado pensando sempre em três camadas (INC, 2009):

a) Usuário;

b) Dados;

c) Coleta e Controle.

A terceira camada é responsável pela coleta e controle dos dados. Nessa camada que existe toda a comunicação com os servidores clientes que estão sendo monitorados. A segunda camada é a responsável pelo armazenamento e processamento dos dados e eventos. Nela que acontece toda a lógica de detecção de erros. A primeira camada é a camada visível para o usuário, em que seu principal elemento é a interface gráfica.

A Figura 3.4 ilustra todas as três camadas e os objetos (plugins) existentes em cada uma delas. É possível observar também qual a função específica de cada plugin.

A detecção dos erros acontece no plugin chamado ZenEvents (INC, 2012). Ele possui uma base de dados, em MySQL, com todas as informações dos eventos já previamente criados. Ao receber e processar as informações dos plugins que coletam os dados, ele armazena em seu próprio banco de dados e, após aplicar a lógica dos eventos, informará o estado de cada um deles.

3.4.3 Análise de Problemas

A interface gráfica do Zenoss possui diversas funcionalidades práticas para auxiliar na visualização e investigação dos problemas. Uma vez que o dispositivo tenha sido mode-

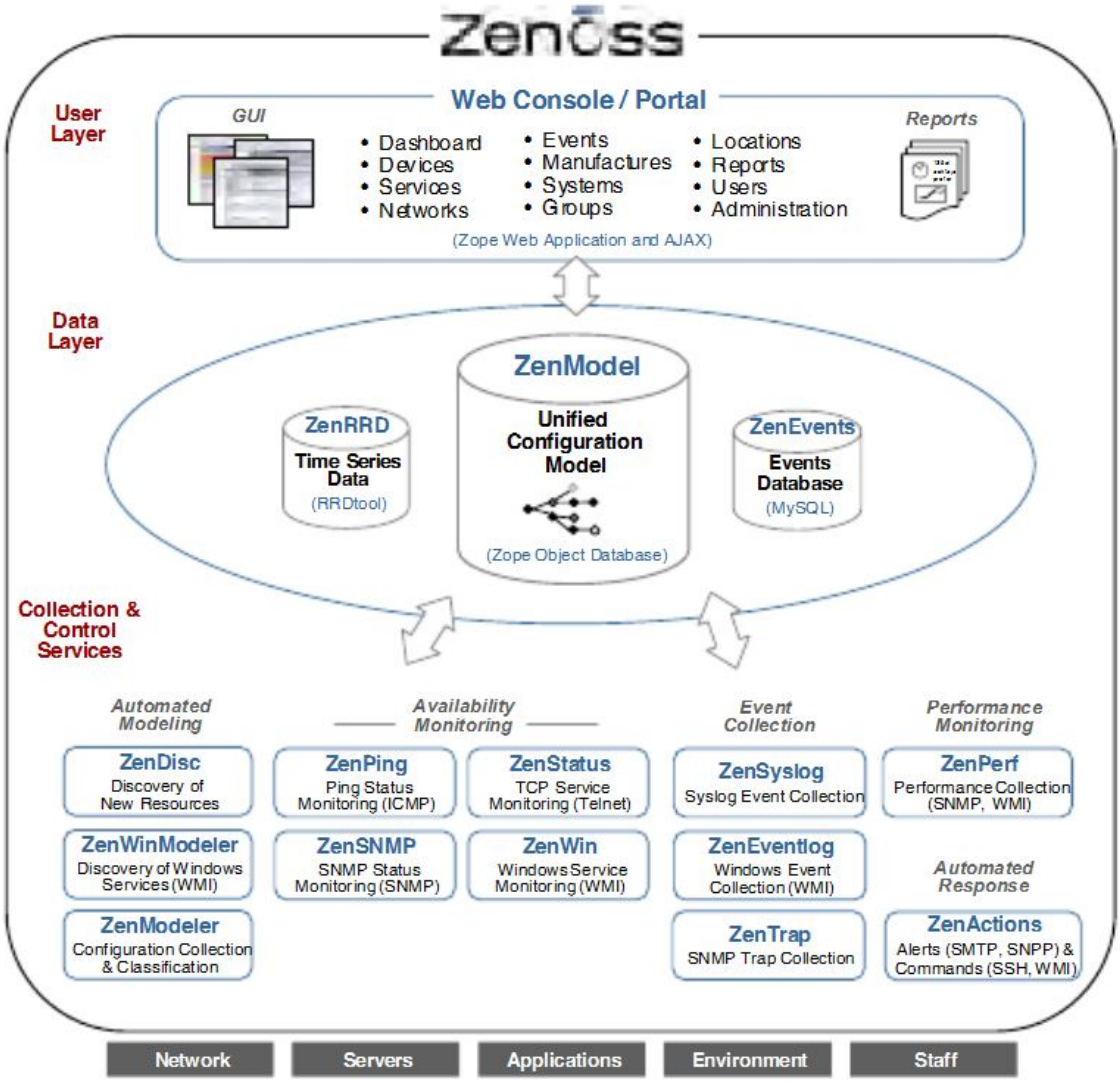


Figura 3.4: Zenoss: Camadas e Arquitetura. Extraído de (LINUX WORLD MAGAZINE, 2006)

lado, existem muitas informações importantes atualizadas online que facilitam qualquer investigação (uso de CPU/memória/disco, usuários conectados, entre outras).

Essa ferramenta também é muito voltada para a parte gráfica. Dessa maneira existem muitos gráficos disponíveis para o usuário acessar. Também existe a possibilidade de serem configurados relatórios periódicos para serem gerados, que podem facilitar a indicação de problemas ao investigar as tendências de comportamento.

No próximo capítulo será feita uma comparação entre essas três ferramentas e como elas se comportam perante uma série de atributos, características e critérios escolhidos.

4 COMPARAÇÃO ENTRE OS SOFTWARES

Este capítulo abordará comparações de uma série de conceitos escolhidos para as ferramentas. O objetivo principal aqui é identificar pontos fracos e pontos fortes de cada uma delas.

Se for possível, identificar oportunidades de melhoria também serão ressaltadas neste capítulo.

4.1 Critérios de Escolha

As seções e subseções abaixo foram escolhidas por terem sido identificadas como os principais atributos de uma ferramenta de monitoração. Foram feitas algumas reuniões com profissionais da área de IT para se chegar a essa lista.

Esses critérios não são específicos de nenhuma aplicação. Eles deveriam ser comuns a todas as ferramentas de monitoria e, nas próximas páginas, será visto que existem muitas oportunidades de melhoria.

4.2 Documentação e Disponibilidade

Uma vez que foram escolhidas ferramentas de código aberto para serem analisadas, é de extrema importância que elas tenham uma documentação completa. Sem isso, poderia ficar muito difícil aproveitar a capacidade de personalização que uma ferramenta de código livre possui.

Aqui será avaliada a documentação oficial sobre cada uma das ferramentas. O que existe nessa documentação (tutoriais, explicação da estrutura do código) e como ela é divulgada (livro, livre na internet, pago na internet). Pontos que também serão considerados é a disponibilidade de treinamentos e a linguagem em que a documentação é liberada.

O Zabbix possui uma ampla gama de artigos tutoriais que são disponibilizados diretamente do site da ferramenta. Toda a documentação também está disponível em cinco idiomas diferentes: Inglês, Francês, Japonês, Português e Russo.

Ainda para o Zabbix, existem diversos cursos que são realizados no mundo todo. Detalhes sobre os treinamentos podem ser observados na Figura 4.1.

Toda a documentação do Nagios está também disponibilizada no site da empresa mas toda ela em formato PDF. Isso acaba dificultando a navegação pela informação pois os artigos encontram-se muito divididos e não existe uma clara explicação do que há em cada um deles. Além disso, não existem muitos artigos sobre configuração do Nagios disponíveis na internet.

Um ponto favorável do Nagios é que existem alguns tutoriais em vídeo disponibili-

Course Title	Zabbix Certified Specialist	Zabbix for Large Environments
Product Covered	Zabbix 2.0 (also applies to 1.6, 1.8)	
Format	Small groups (up to 15 people)	
Duration	3 days	2 days
Requirements	Advanced computer literacy and knowledge of operating systems	Zabbix Certified Specialist certificate
Completion Certificate	Zabbix Certified Specialist Certificate	Zabbix for Large Environments Certificate
Price*	€745.00 / $1,035.00	€715.00 / $995.00

** Indicated price is for training courses organized by Zabbix. Zabbix reserves the right to change prices without prior notice. Price for training course organized by Zabbix Training Partners may vary.*

Figura 4.1: Zabbix: Detalhes sobre os Cursos. Extraído de (ZABBIX, 2012)

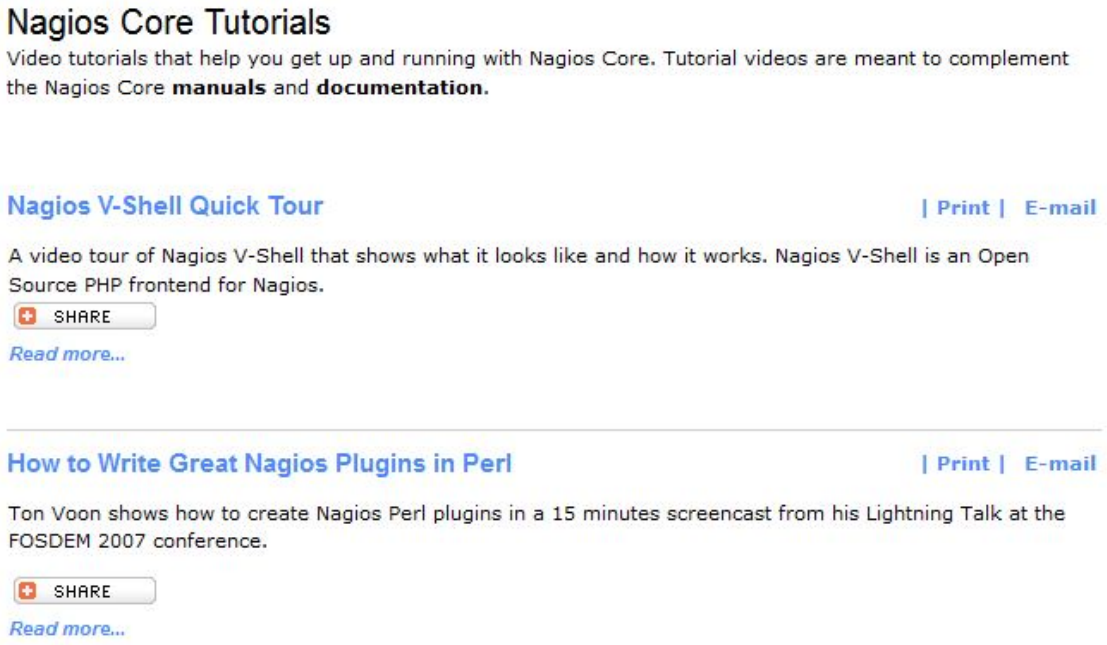


Figura 4.2: Nagios: Detalhes sobre os vídeos tutoriais. Extraído de (NAGIOS, 2012)

zados com informação sobre como escrever novas partes de código e uma rápida revisão sobre a ferramenta. Eles estão exemplificados na Figura 4.2.

Já o Zenoss possui uma documentação bastante atualizada sobre a ferramenta. A documentação oficial é baseada em um arquivo, também no formato PDF, que traz todas as informações sobre aquela versão da ferramenta. Entretanto, não há uma abordagem direta quanto a problemas que podem ocorrer durante os processos executados.

Além disso, não existem tutoriais detalhados oficiais sobre o Zenoss, nem treinamentos abertos ao público. Tutoriais são muito importantes para pessoas começando a usar as ferramentas para que seja possível usar todos os benefícios que a ferramenta dispõe.

A documentação mais completa encontrada foi a do Zabbix, inclusive disponibilizando treinamentos ao redor do mundo. O Nagios possui diversos artigos tutoriais que facilitam instalações e configurações. Sem dúvida a documentação do Zenoss não é muito aprofundada e dificulta o uso da ferramenta.

Tabela 4.1: Linguagens de programação utilizadas

Ferramenta	Linguagem de Programação
Zabbix	C e PHP
Nagios	Perl
Zenoss	Python e Zope

4.2.1 Código Fonte

O objetivo de uma ferramenta ter seu código fonte disponibilizado gratuitamente é que ela possa ser modificada e analisada com mais eficácia. Para isso, é necessário entender como o código fonte está estruturado, quais linguagens foram utilizadas no desenvolvimento da ferramenta, entre outros atributos. A análise feita de agora em diante não entrará em detalhes sobre classes, funções e atributos do código. Será uma análise mais estrutural.

A ferramenta Nagios é sem dúvida a que mais facilita mudanças de código. Ela é estruturada de uma maneira que novos plugins podem ser desenvolvidos sem a necessidade de modificar os antigos. Os serviços e monitores novos a serem configurados são feitos para chamar uma linha de comando de um executável, passando alguns parâmetros conforme necessário. Dessa maneira, para se criar novos tipos de monitores basta desenvolvê-los seguindo os padrões documentados.

O código fonte do Nagios foi desenvolvido em Perl. Plugins do Nagios podem ser desenvolvidos em qualquer linguagem de programação pois a ferramenta só está interessada na saída do programa. Entretanto, algumas linguagens são recomendadas:

- C;
- Perl;
- Python;
- Shell;
- Scripts.

Toda a estrutura do Zabbix é desenvolvida em C e PHP. A estrutura do código fica mais escondida e não é tão aberta quanto a do Nagios mas é possível se encontrar todos os arquivos e manipulá-los uma vez que se tenha um conhecimento intermediário de C. Diferentemente do Nagios, a cada mudança no código a aplicação deve ser compilada novamente. No Nagios basta reiniciar o serviço e as mudanças se adaptarão (a não ser que seja uma mudança estrutural).

Já o Zenoss foi a ferramenta que mostrou menos facilidade de adaptação. O código aparece muito espalhado, em diversos arquivos e existem muitas interconexões (chamadas entre os arquivos) que não estão documentadas claramente. As linguagens de programação utilizadas são o Python, majoritariamente, e o Zope.

A Tabela 4.1 traz uma comparação entre as linguagens de programação utilizadas:

4.2.2 Fóruns

Os fóruns são ferramentas de auxílio extremamente úteis. Usuários e administradores trabalham juntos para ajudar outros usuários e fazer com que todos possam utilizar melhor as ferramentas.

Tabela 4.2: Endereço dos fóruns oficiais

Ferramenta	Fórum
Zabbix	http://www.zabbix.com/forum/
Nagios	http://support.nagios.com/forum/
Zenoss	http://community.zenoss.org/community/forums

Tabela 4.3: Tempo de resposta, em horas, dos fóruns oficiais

	Zabbix	Nagios	Zenoss
Perguntas Respondidas	7	10	9
Resposta mais rápida	2,6	0,4	0,05
Resposta mais demorada	1134,65	99,55	6223
Tempo médio de respostas	229,1	6,45	39,35

Foram analisados os fóruns oficiais das ferramentas, que podem ser observados na tabela 4.2.

Foram analisadas as dez últimas perguntas de cada um deles e foi analisado o tempo médio, em horas, até a primeira resposta. Os resultados encontrados podem ser observados na tabela 4.3. Foram removidos dos cálculos de média os pontos mais rápidos e mais lentos de cada ferramenta.

A partir dessa análise, podemos observar claramente que o fórum com participação mais assídua dos usuários é o fórum da ferramenta Nagios. Todos os critérios avaliados foram melhores no fórum dessa ferramenta.

Quanto às outras ferramentas, o site do Zenoss leva uma vantagem sobre o do Zabbix. A maioria das respostas avaliadas foram respondidas e a média, sem considerar os pontos extremos, ficou muito menor que a do Zabbix.

O fórum do Zabbix teve o pior resultado: algumas perguntas não respondidas (representando 30% do total), situações extremas longe do ideal e média muito alta.

4.2.3 Suporte

Um outro aspecto bastante importante de ser avaliado é a disponibilidade de suporte oficial. Fóruns são apropriados mas, em alguns casos, é necessário um auxílio mais especializado. Esses casos serão avaliados agora.

Nas empresas Zabbix e Zenoss foi disponibilizada mais de uma opção de suporte. Nesse caso, a fim de comparar pacotes de mesmo nível comercial, foram escolhidos os pacotes mais avançados de suporte.

Foram selecionados alguns dos tipos de suporte considerados mais importantes para efetuar a comparação. Eles podem ser observados na tabela 4.4. A Figura 4.3 mostra mais detalhes sobre os tipos de suporte disponível para cada ferramenta.

Foi possível observar que as empresas Zabbix e Zenoss possuem um suporte muito qualificado. O suporte disponível pela empresa Zabbix é mais completo que o das outras duas, chegando até a disponibilizar atendimento direto no prédio da empresa ou cliente.

A empresa Zabbix também disponibiliza treinamento profissional para administradores e usuários. Quanto ao Zenoss, a opção de treinamento não está incluída no pacote e é apenas para administradores.

Não existe nenhum tipo de suporte mais qualificado para o Nagios. Todo o suporte é feito por contato telefônico ou digital. Não há disponibilidade de treinamento nem

Figura 4.3: Tabelas ilustradas sobre opções de suporte disponíveis

Support Tier Offerings

	Bronze	Silver	Gold	Platinum	Enterprise
Number of incidents	4	8	Unlimited	Unlimited	Unlimited
Support availability times (h-d)	8 x 5	8 x 5	8 x 5	24 x 7	24 x 7
Guaranteed response times *	2 Days	1 Day	4 Hours	4 Hours	4 Hours
Online case submission	✔	✔	✔	✔	✔
Phone technical support		✔	✔	✔	✔
Standard Zabbix builds		✔	✔	✔	✔
Remote Troubleshooting			✔	✔	✔
Zabbix back-up tool			✔	✔	✔
Distributed monitoring with Zabbix Proxy			✔	✔	Unlimited
Emergency response within 90 minutes				✔	✔
Performance Tuning				✔	✔
Pre-compiled software according to customer request				✔	✔
Assigned primary support contact					✔
On-site visit **					✔
On-site professional training ***					✔
Upgrade to the latest version					✔
Environment Reviews					✔
Custom Zabbix builds					✔
Sponsored development priority					✔

Technical Support Terms and Conditions

Request a quote > Contact Sales > Request a call back > Sign up for a Demo >

* *While we offer specific timeframes within our support service level guarantees, every effort will be made to respond more quickly when resources are available.*

** *One 5 business day visit to a customer office by a leading consultant of Zabbix, which may be used to receive consultations, perform tuning of Zabbix installation, deliver additional training etc.*

*** *Zabbix Certified Specialist and Zabbix for Large Environments training at a customer location for up to 5 customer's employees. According to a customer request a standard program can be adjusted to cover topics selected by a customer.*

(a) Zabbix. Extraído de (ZABBIX, 2012)

	Gold	Platinum
Product Support		
Support Hours	Mon-Fri, 7AM to 8PM EST	
Web-based portal	Unlimited	Unlimited
E-mail	Unlimited	Unlimited
Phone	Unlimited	Unlimited
Priority 1/High Severity Response Time SLA	4 hours	2 hours
Remote Trouble Shooting	✓	✓
24/7 Critical Issue Support		✓
Maintenance & Administration		
Patches	✓	✓
Upgrades	✓	✓
New Releases	✓	✓
Training		
Administrator Training - Web-based	$1500/seat	$1500/seat
Advanced User Training	$2000/seat	$2000/seat
Professional Services		
Quick Start Package	$15k	$15k
Custom Professional Services	$250/hr	$250/hr
Planning/Deployment	$250/hr	$250/hr
Application Customization	$250/hr	$250/hr
Custom Programming	$300/hr	$300/hr

(b) Zenoss. Extraído de (ZENOSS, 2012)

Tabela 4.4: Disponibilidade dos Tipos de Suporte

	Zabbix	Nagios	Zenoss
Suporte Online	Sim	Sim	Sim
Suporte Telefônico	Sim	Sim	Sim
Auxílio em Atualizações/Instalações	Sim	Não	Sim
Auxílio Emergencial	Sim	Não	Sim
Visita na empresa	Sim	Não	Não
Treinamento	Sim	Não	Não

tratamento de problemas. Além disso, contato telefônico só é possível durante 8 horas dos dias de semana, enquanto que nas outras empresas esse tipo de contato era constante.

4.2.4 Instalação

Ferramentas de código aberto devem ter a instalação facilitada com tutoriais uma vez que existem muitas variáveis envolvidas (código, variáveis de sistema, pacotes adicionais, entre outros).

A instalação mais fácil de ser feita é, sem dúvida alguma, a do Nagios. Em um sistema operacional Ubuntu, o pacote do Nagios está disponível no sistema para download e instalação. Basta baixá-lo e instalá-lo no servidor e no cliente (pacotes diferentes) e tudo estará pronto. Todos os pacotes adicionais são instalados junto e todas as configurações também são ajustadas durante a instalação.

Após a instalação, basta iniciar a interface web e configurar o usuário. Para novos monitores, devem ser modificados os arquivos de configuração. Para toda a configuração e instalação do Nagios existem ótimos tutoriais disponíveis na internet.

Instalar e configurar o Zabbix também não é uma tarefa muito complicada. Basta seguir os tutoriais oficiais disponíveis no site oficial e tudo irá correr bem. O único problema encontrado foi alguma incompatibilidade de versões, mas que foi resolvida rapidamente seguindo outros tutoriais disponíveis no site.

O caso mais problemático foi o do Zenoss. Instalá-lo não é uma tarefa fácil. Uma máquina virtual com o servidor Zenoss é disponibilizada pronta, mas ela não é compatível com os objetivos desse estudo (instalar diretamente a ferramenta). Sem essa máquina virtual, a segunda opção é utilizar um script de instalação feito para o sistema operacional Red Hat, que também não satisfaz os requisitos do trabalho (sistema operacional Ubuntu).

Não sendo possível seguir a instalação pelos meios convencionais, foi necessário buscar ajuda em fóruns. Foram encontrados alguns scripts de instalação mas nenhum deles se mostrou completo. A saída encontrada foi juntar partes dos scripts para concluir a instalação com êxito.

Durante a execução desses scripts, foram encontrados muitos erros inesperados. A figura 4.4 ilustra bem o descrédito que os usuários têm pela facilidade da instalação. Após conseguir resolver um problema de instalação, um usuário diz "Esperando para ver onde irá falhar em seguida".

Após a instalação terminar, a configuração da interface também se mostrou complicada. A documentação oficial existente não leva em conta que o usuário pode encontrar falhas. Quando elas ocorrem, mais uma vez deve-se recorrer a métodos não oficiais (fóruns que não são do Zenoss).

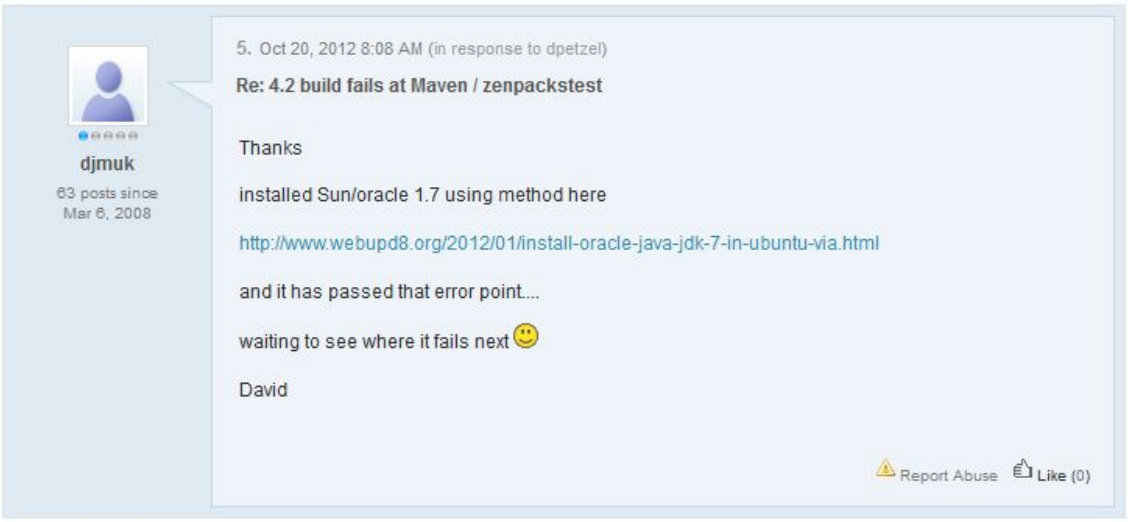


Figura 4.4: Zenoss: Exemplo de descrédito pelos scripts de instalação. Extraído de (ZENOSS, 2012)

Tabela 4.5: Tabela parcial de comparação

	Zabbix	Nagios	Zenoss
Documentação e Disponibilidade	3	2	1
Código Fonte	2	3	1
Fóruns	1	3	2
Suporte	3	1	2
Instalação	2	3	1

4.2.5 Tabela Parcial

Terminada uma primeira fase de análises, a Tabela 4.5 mostra um resumo do que foi identificado nas ferramentas até aqui. É possível observar que a ferramenta Zenoss foi a que teve o pior desempenho nessa fase inicial. A ferramenta Nagios leva uma pequena vantagem sobre a Zabbix nessa primeira parte por possuir uma maior variedade de opções.

4.3 Interface com Usuário

Outro aspecto muito importante de uma ferramenta de monitoração é a sua capacidade de mostrar aquilo que é monitorado para o usuário de uma forma que ele entenda. Uma ferramenta pode ser a mais complexa possível, mas se ela for difícil de ser usada, não tiver uma interface gráfica boa, ela não será usada.

Nos próximos parágrafos, será vista uma análise da interface entre as ferramentas e os usuários/administradores. Não será avaliada apenas a parte gráfica, toda a interação com os usuários (seja ele final, seja ele moderador) será levada em conta.

4.3.1 Criação de Monitores

A criação de monitores pode ser feita de diferentes maneiras nas ferramentas de monitoria. As ferramentas deste estudo se enquadram em três categorias diferentes:

1. Arquivos de Configuração;

Object	Description
host	Hosts are physical devices like servers, routers, and firewalls.
hostgroup	Host groups are collections of hosts that generally have something in common, like their type or location.
service	Services run on hosts and can include actual services like SMTP or HTTP, or metrics such as disk space.
servicegroup	Service groups are collections of services that generally have something in common.
contact	Contacts are escalation or notification points that can potentially be contacted when an event occurs.
contactgroup	Contact groups are collections of contacts. All contacts need to be in a contact group.
timeperiod	Time periods are defined windows of time—for example, during business hours.
command	Commands are called by the check process to perform an action—for example, a command to check the status of a host using the ping command.
servicedependency	Allows a service or services to be dependent on other services.
serviceescalation	Provides a notification escalation process for services.
hostdependency	Allows a host or hosts to be dependent on other hosts.
hostescalation	Provides a notification escalation process for hosts.
hostextinfo	Host Extended Information changes and customizes the way hosts are displayed on the Nagios web console.
serviceextinfo	Service Extended Information changes and customizes the way services are displayed on the Nagios web console.

Figura 4.5: Nagios: Possíveis objetos a serem criados. Extraído de (TURNBULL, 2006)

2. Interface Gráfica usando modelos prontos;

3. Interface Gráfica e estrutura de classes.

A ferramenta Nagios se enquadra no tipo de criação de monitores utilizando arquivos de configuração. A ideia principal que deve ser entendida para criar monitores para o Nagios é que todas as chamadas são chamadas do terminal. Isso quer dizer que todo o monitor implementado pode ser testado e ajustado usando o terminal de linhas de comando do Linux.

Uma vez que o usuário, ou administrador, sabe exatamente o que deseja que o monitor procure, basta procurar se existe um plugin pronto para isso. Essa informação está disponível no site oficial da empresa. Se não houver nenhum plugin disponível, existe a opção de tentar baixar algum outro pacote que faço esse trabalho. Se nada disso funcionar, o usuário pode desenvolver um plugin para fazer o que ele deseja.

A configuração dos arquivos segue os padrões da empresa e está muito bem documentado nos tutoriais. No Nagios, um monitor é chamado de serviço. A figura 4.5 mostra todos os objetos de definição que podem ser criados dentro dos arquivos de configuração Nagios. Como eles, toda a hierarquia de servidores e monitores é construída.

O Zenoss se enquadra no tipo de criação de monitores que utiliza modelos prontos via interface gráfica. Aqui segue-se o conceito de classe de eventos, em que um evento pode herdar atributos de outros (por exemplo, o plugin utilizado). A instalação padrão do Zenoss também instala o pacote de plugins do Nagios. A diferença reside na forma que esses monitores são criados. A figura 4.6 mostra a criação de um novo monitor (chamado de evento).

Um fato importante de ser observado aqui é que se o tipo de monitoração esperada não é encontrada em nenhum dos modelos prontos, deve-se desenvolver um novo modelo.

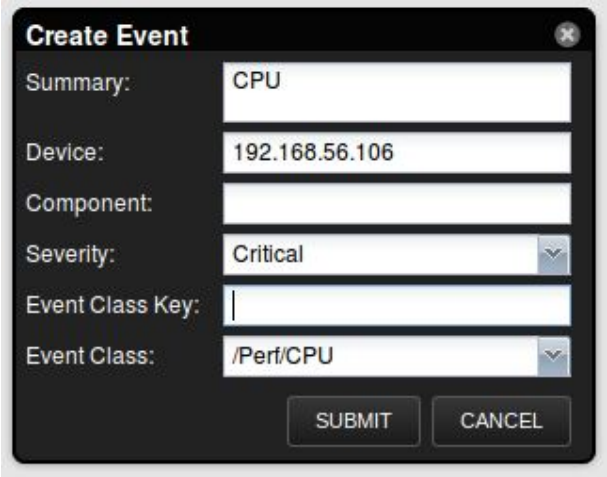


Figura 4.6: Zenoss: Criação de um novo monitor usando a interface

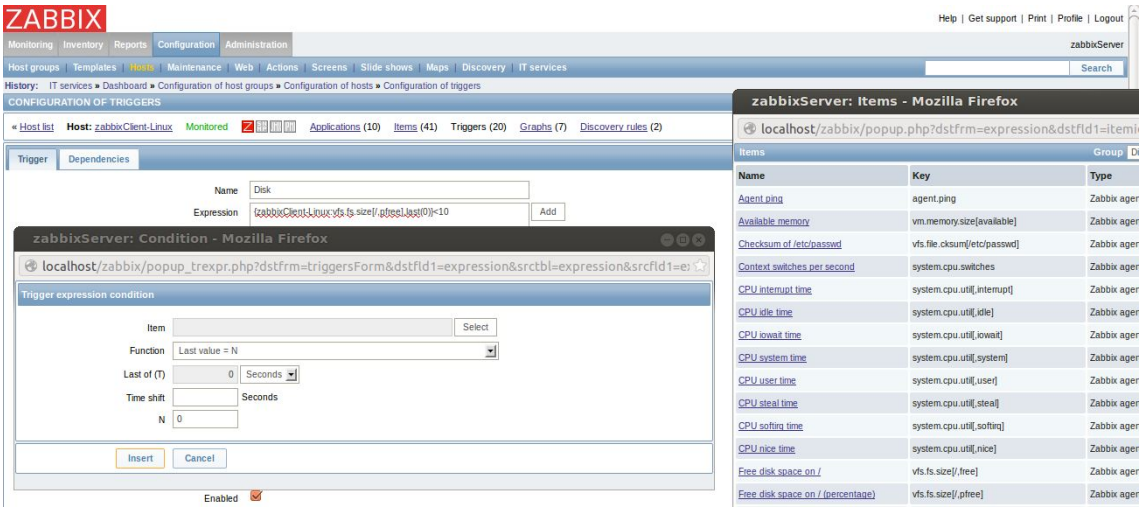


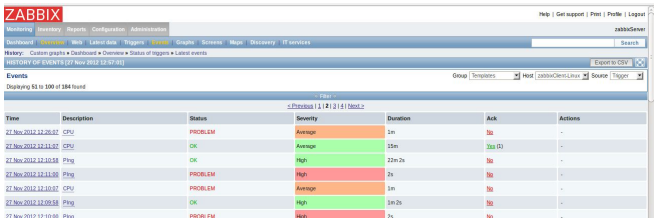
Figura 4.7: Zabbix: Criação de um novo monitor usando a interface

Isso é uma tarefa mais complicada com essa ferramenta pois envolve uma interação maior com outras partes do software.

O Zabbix possui esse mesmo problema. Ele se enquadra no tipo de criação de monitor utilizando a interface gráfica. É usada uma estrutura semelhante à de chamadas de classe, o que pode ser observado na Figura 4.7.

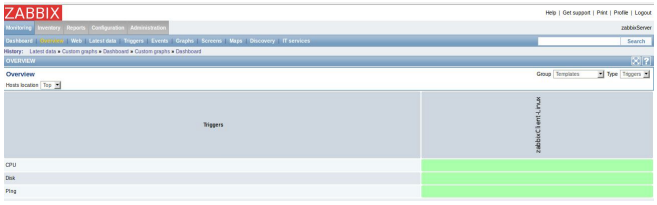
A criação mais fácil da monitoração foi a efetuada na ferramenta Zabbix. A interface facilita bastante a criação e modificação de monitores com o uso do Construtor de Expressões, uma interface gráfica criada diretamente para auxiliar na criação dos monitores (chamados de Disparadores). O Nagios também se mostrou bastante simples na criação dos monitores. Fica em desvantagem pelo fato de não possuir uma interface gráfica para isso. O Zenoss mais uma vez se mostrou com bom potencial, mas muito complicado pois a criação dos novos eventos deveria acontecer em muitos lugares diferentes para fazer tudo funcionar corretamente. Ainda, as interfaces encontradas não eram fáceis de serem usadas.

Figura 4.8: Zabbix: Visualização do estado dos monitores

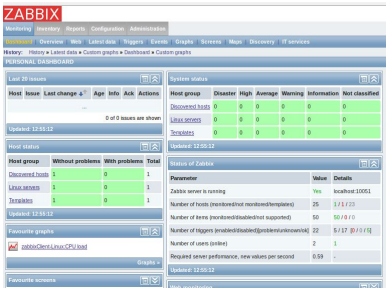


(a) Eventos

(b) Disparadores



(c) Visão Geral



(d) Dashboard

4.3.2 Estado dos Monitores

A chamada monitoração via Dashboard é muito importante em empresas de monitoria. Além de ser notificado quando um monitor identificar um problema, os usuários também devem conseguir visualizar o estado dos monitores a qualquer momento. Esse é objetivo do Dashboard: visualizar o estado de todos os monitores criados, de uma maneira prática e rápida.

Esta subseção abordará como os monitores podem ser visualizados na interface. Diferenciação de quais estão alertando, estado atual de cada monitor (inclusive os que estão concluindo que não há nada errado) e possibilidade de rodar testes a qualquer hora são atributos desejáveis.

A ferramenta Zabbix possui uma visão bastante precisa do estado de todos os monitores. Podem ser visualizados os monitores por servidor, por hierarquia de servidores (aspecto muito importante quando se possui uma rede grande), por hierarquia de monitores (também importante quando se possui uma rede grande com muitos monitores) e também podem ser visualizados todos os monitores configurados.

A Figura 4.8 mostra algumas das principais telas do Zabbix em que é possível visualizar o estado dos monitores. É possível observar que podemos ter diversas visões diferentes para o mesmo monitor, o que facilita bastante a utilização da ferramenta.

Uma funcionalidade bastante interessante do Zabbix é testar o estado atual de um monitor a qualquer hora. Se um monitor falhar e o usuário achar que resolveu o problema, ele pode forçar um novo teste do monitor para ter certeza que ele ficará bem.

Outro aspecto importante do Zabbix é a possibilidade de visualizar os últimos dados coletados. Dessa maneira é possível verificar que tudo está sendo coletado corretamente

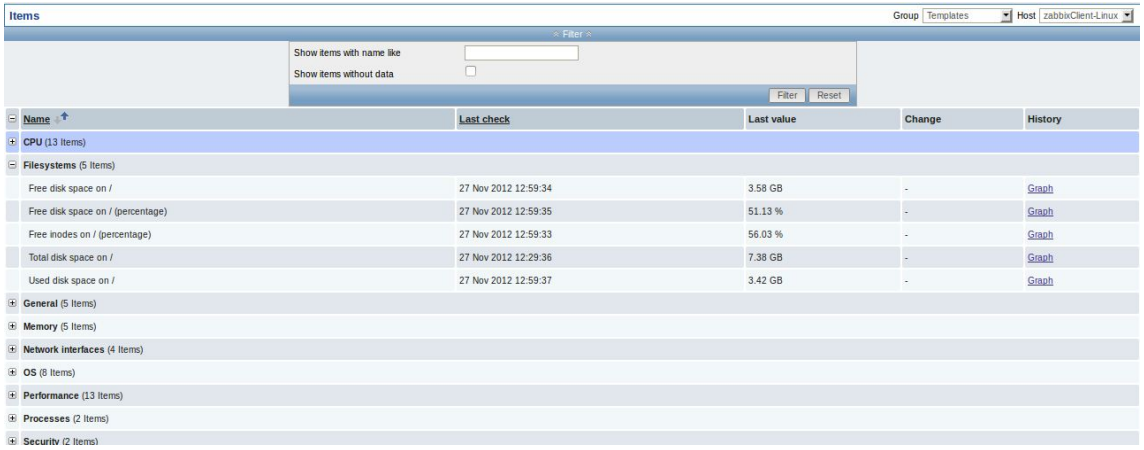


Figura 4.9: Zabbix: Visualização dos últimos dados coletados

e ainda verificar se os últimos dados estão de acordo com o esperado. Isso pode ser observado na Figura 4.9.

Já no software Nagios, existe uma chamada Visão Tática que mostra quantos servidores existem em cada estado. Um servidor pode estar em algum dos seguintes estados:

- Fora do Ar (Down);
- Inatingível (Unreachable);
- Funcionando (Up);
- Pendente (Pending).

Existe também uma visão de todos serviços e agrupamento dos mesmos por estados. Um serviço, ou monitor, pode ser classificado em:

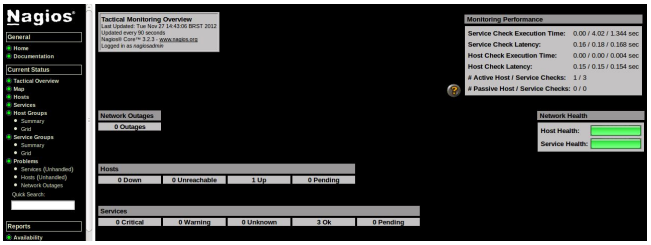
- Crítico (Critical);
- Advertência (Warning);
- Desconhecido (Unknown);
- OK;
- Pendente (Pending).

Ao se procurar mais detalhes sobre cada um dos monitores, é possível obter visões separando-os por estado, por servidor e por grupo de servidores. A Figura 4.10 mostra algumas das principais telas do Nagios.

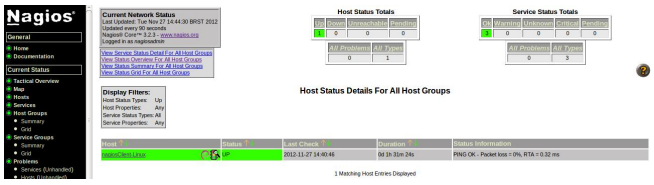
Assim como no Zabbix, o Nagios também disponibiliza a funcionalidade de se testar um monitor a qualquer momento, facilitado assim a investigação. Além disso, é possível re-agendar a próxima vez que o monitor irá coletar dados. Isso pode ser útil ao se trabalhar com problemas de duração prolongada.

Na ferramenta Zenoss, não existe uma maneira prática de visualizar o estado dos monitores que não estão alertando. Existem algumas visualizações diferentes disponíveis quando se deseja acessar os eventos (monitores, serviços) que estão com problema. Elas estão ilustradas na Figura 4.11. Existe a possibilidade de acessar relatórios que trazem os

Figura 4.10: Nagios: Visualização do estado dos monitores

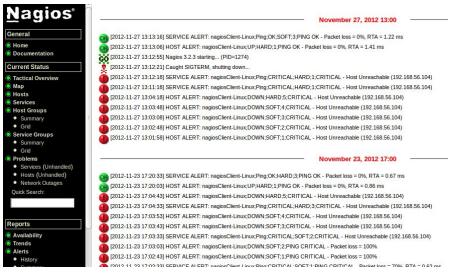


(a) Visão Tática



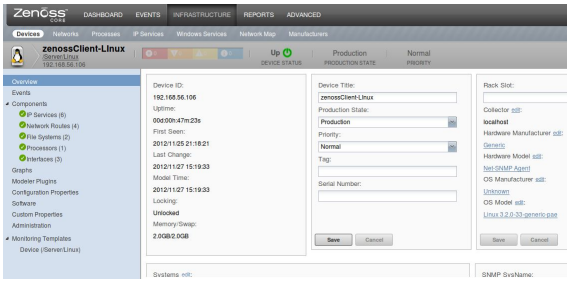
(b) Servidores

(c) Monitores

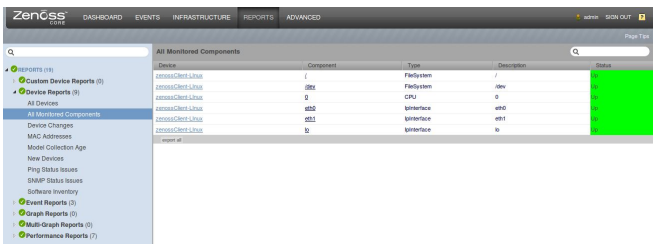


(d) Histórico de Alertas

Figura 4.11: Zenoss: Visualização do estado dos monitores

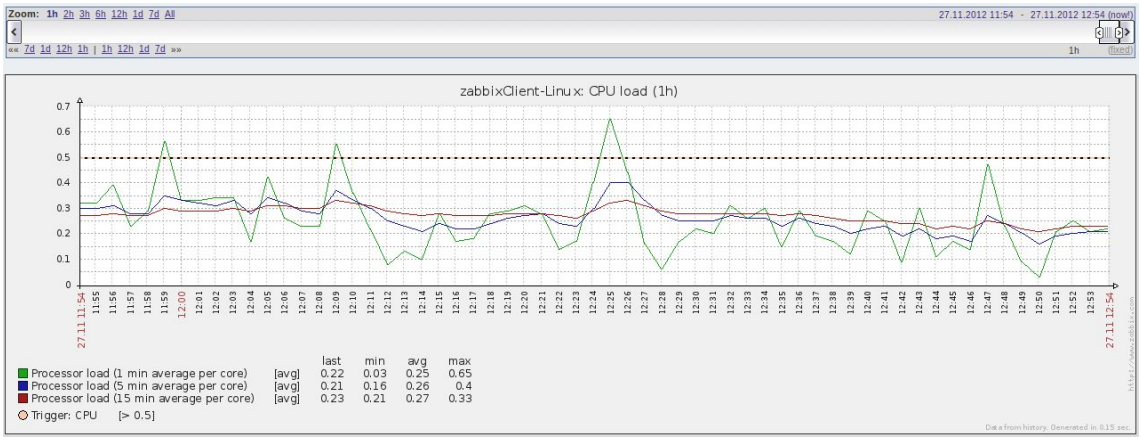


(a) Visão com detalhes de um servidor

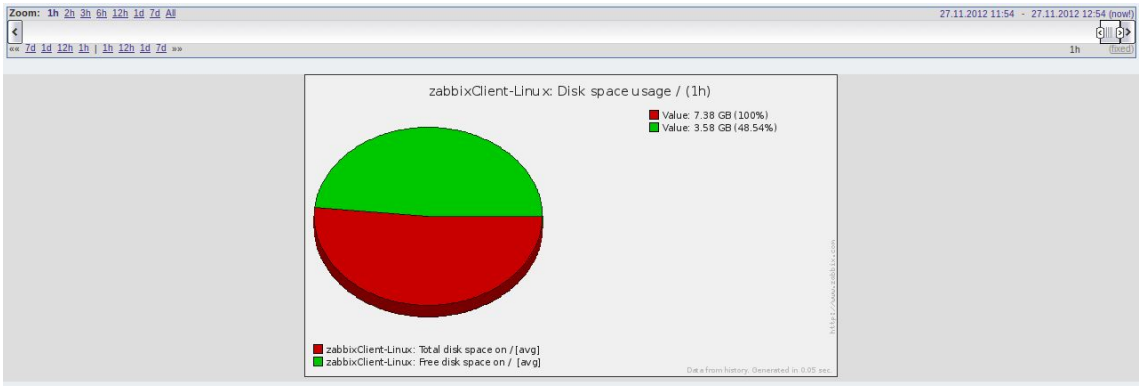


(b) Estado dos monitores

Figura 4.12: Zabbix: Gráficos de análise



(a) Gráfico de uso CPU



(b) Gráfico de percentual de utilização de disco

estados dos monitores. Esse tipo de relatório ajuda a visualizar o estado dos monitores, apesar de não trazer muitas informações.

Sem dúvida nenhuma, a comparação das três ferramentas coloca o Zabbix em primeiro lugar pela facilidade de uso da interface. O Zenoss possui um alto potencial, mas não é fácil de ser usado. Por isso, a segunda melhor ferramenta é o Nagios.

4.3.3 Gráficos de Análise

Uma outra funcionalidade que aumenta muito a capacidade analítica das ferramentas é a capacidade de geração gráfica. Gráficos sobre os estados dos monitores, gráficos de análise de comportamento de um determinado monitor e outros tipos de gráficos são muito importantes para uma ferramenta de monitoração. Aqui, será realçado como cada uma das ferramentas gera gráficos para auxiliar a análise de problemas.

A ferramenta Zabbix possui uma visão gráfica bastante utilizada. É possível gerar gráficos de comportamento de valor de qualquer tipo de monitor. Os gráficos armazenam o valor de um determinado monitor em determinado momento. A Figura 4.12 mostra alguns tipos de gráficos disponíveis no Zabbix.

Uma funcionalidade muito importante disponível no Zabbix é a capacidade de gerar gráficos históricos. Isso permite uma análise de comportamento que ajuda na hora de se analisar problemas.

O Nagios não possui uma grande capacidade gráfica. É possível sim gerar gráficos de

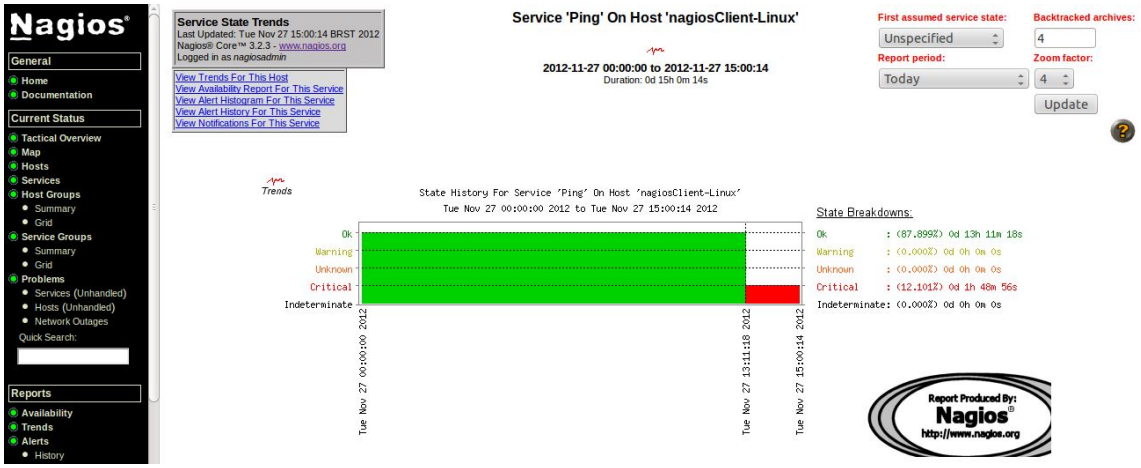
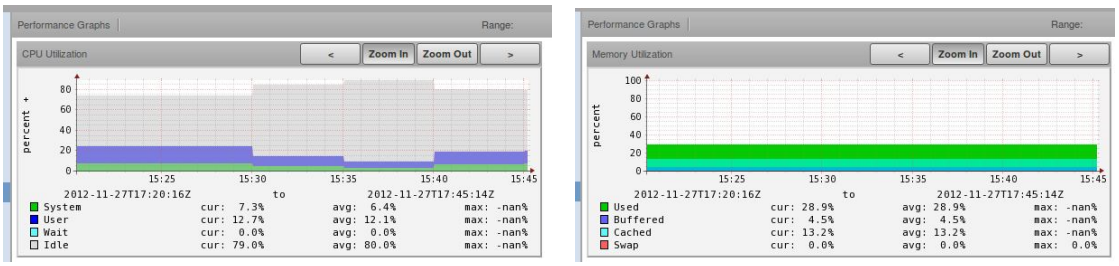


Figura 4.13: Nagios: Gráfico de análise

Figura 4.14: Zenoss: Gráficos de análise



(a) Gráfico de uso CPU (b) Gráfico de percentual de utilização de memória

análise para ele mas eles são um pouco mais complicados de serem gerados do que nas outras ferramentas. Além disso, os gráficos gerados são um pouco menos claros.

O Zenoss é uma ferramenta muito boa para gerar alguns gráficos de análise específicos. Existem alguns gráficos pré-definidos que são gerados uma vez que um dispositivo (servidor) é adicionado à ferramenta. A Figura 4.14 ilustra alguns desses gráficos.

Os quatro tipos de gráficos enumerados abaixo são os gráficos padrões do Zenoss:

- Média de Carga (Load Average);
- CPU Utilization (Utilização de CPU);
- Memory Utilization (Utilização de Memória);
- Entrada e Saída (IO);

Se o usuário desejar gerar algum outro tipo de gráfico, é necessário modificar o código da ferramenta para adicionar esses gráficos. Entre outras palavras, não é uma tarefa fácil.

4.3.4 Relatórios

A geração de relatórios customizados é facilitada por ferramentas manipuladoras de dados, como o Microsoft Office Excel, LibreOffice Calc, entre outras. O ponto comum entre as principais ferramentas voltadas à analise de dados é a existência de tabelas de formatação e de dados.

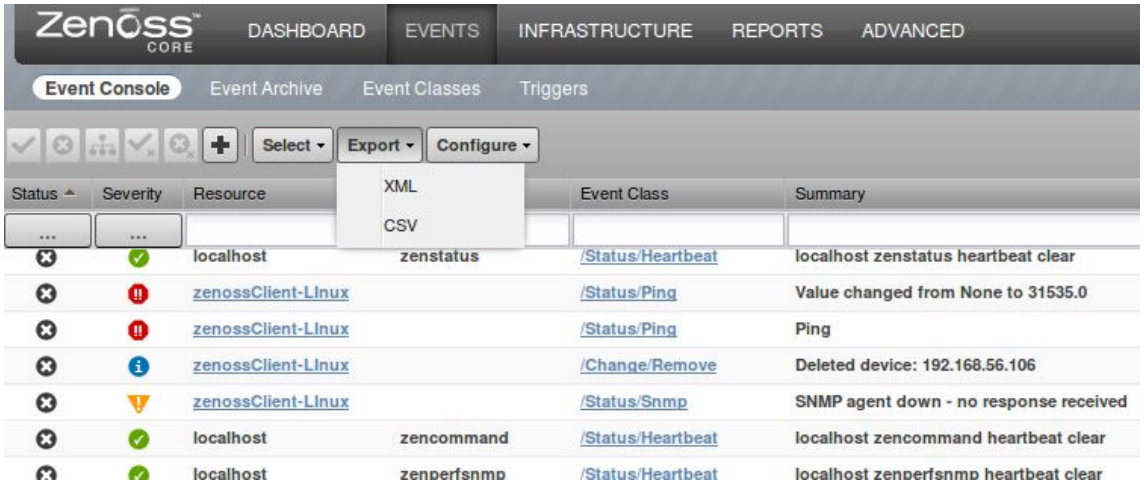


Figura 4.15: Zenoss: Opções de exportar dados

É bastante normal surgir a necessidade de relatórios diferenciados dos já existentes. Para que isso possa ser feito, existe a necessidade de extrair os dados da ferramenta de monitoração em um formato que as ferramentas como o Excel entendam.

A ferramenta Zabbix não fornece compatibilidade para exportar os dados diretamente no formato CSV, por exemplo. Entretanto, é possível visualizar os dados em formato de tabela na tela e, a partir disso, colar para uma ferramenta de manipulamento de dados.

A interface do Nagios não possibilita nenhuma criação de relatórios desse tipo. Não é possível visualizar os dados em formato de tabela nem extrair diretamente no formato CSV, por exemplo. Para se gerar relatórios de Excel, deve-se configurar plugins que irão buscar essas informações no banco de dados do Nagios e lidar com a compatibilidade.

O Zenoss é a única ferramenta que traz alguma compatibilidade desejada. Na área de manipulação de eventos, é possível extrair os dados no formato CSV ou XML. Essas opções podem ser observadas na Figura 4.15.

4.3.5 Usabilidade Geral

Aqui será abordada a satisfação do usuário ao navegar pela interface dessas três ferramentas.

Foram escolhidas cinco pessoas para auxiliar nesse experimento. Depois de ter todas as ferramentas funcionando plenamente, com todos os monitores configurados, foi pedido para que cada uma das pessoas navegasse pelas ferramentas e analisar os itens a seguir (com notas de 1 a 5, com escala crescente de satisfatoriedade):

- Qualidade dos gráficos;
- Navegação na interface;
- Informações sobre servidor;
- Informações sobre monitor;
- Entender funcionamento de monitor.

As cinco pessoas que desenvolveram o experimento trabalham na área da computação mas nenhuma delas tinha tido nenhum contato com nenhuma dessas ferramentas. Após

Tabela 4.6: Experimento de usabilidade

	Zabbix	Nagios	Zenoss
Qualidade Gráfica	4,2	1,6	4,6
Navegação Livre	4,4	3,4	3,6
Informação sobre Servidor	3	4	4,8
Informação sobre Monitor	2,6	3,8	1,8
Funcionamento de Monitor	1,6	2,4	1

a conclusão dos experimentos, foi possível chegar à Tabela 4.6 que traz alguns aspectos interessantes.

O Nagios e o Zenoss tiveram dois itens em que cada um deles foi eleita a melhor ferramenta. Pode-se observar a clara distinção entre as ferramentas. Nagios foi melhor nos aspectos que envolvem o compartilhamento de informação, enquanto que o Zenoss foi o melhor no que envolve a parte gráfica.

Um aspecto relevante observado foi que o Zenoss teve um desempenho muito bom na informação sobre servidores. Isso acontece porque, ao configurar um servidor nessa ferramenta, ela modela o dispositivo e descobre diversos tipos de informações sobre ele, como quantidade de memória, sistema operacional, placas de rede, entre muitos outros.

O Zabbix mostrou ser a ferramenta mais prática de ser usada, ao ser a melhor ferramenta em navegação livre.

4.4 Capacidade

Até o final deste capítulo, será feita uma análise sobre a capacidade das ferramentas escolhidas. Será mostrado um estudo um pouco mais detalhado sobre em que situações essas ferramentas funcionam corretamente, que tipos de alertas elas podem emitir e como cada usuário pode personalizar a sua interface.

4.4.1 Compatibilidade

Antes de escolher uma ferramenta de monitoramento para utilizar em um sistema, é preciso saber se a ferramenta funciona corretamente nesse sistema. Neta subseção será avaliada a compatibilidade das ferramentas com os pacotes necessários, com o sistema operacional do servidor e do cliente.

O sistema operacional que se mostrou preferencial entre todas as ferramentas foi o sistema Red Hat. Todos os softwares também mostraram preferência por sistemas operacionais do tipo UNIX. A Figura 4.16 mostra os sistemas operacionais divulgados que são suportados oficialmente.

No caso do Zenoss, os seguintes sistemas operacionais são suportados:

- Linux: Red Hat Enterprise;
- Linux: Fedora;
- Linux: Debian;
- Linux: Ubuntu;
- Linux: SUSE;

Figura 4.16: Sistemas Operacionais Suportados

Platform	ZABBIX Server	ZABBIX Agent
AIX	Supported	Supported
FreeBSD	Supported	Supported
HP-UX	Supported	Supported
Linux	Supported	Supported
Mac OS X	Supported	Supported
Novell Netware	-	Supported
Open BSD	Supported	Supported
SCO Open Server	Supported	Supported
Solaris	Supported	Supported
Tru64/OSF	Supported	Supported
Windows NT 4.0, Windows 2000, Windows 2003, Windows XP, Windows Vista	-	Supported

(a) Zabbix. Extraído de (ZABBIX, 2012)

(b) Nagios. Extraído de (NAGIOS, 2012)

Software	Version	Comments
Apache	1.3.12 or later	
PHP	5.0 or later	
PHP modules: **php-gd**	GD 2.0 or later	PHP GD module must support PNG images.
PHP TrueType support		--with-ttf
PHP bc support		php-bcmath, --enable-bcmath
PHP XML support		php-xml or php5-dom, if provided as a separate package by the distributor
PHP session support		php-session, if provided as a separate package by the distributor
PHP socket support		php-net-socket, --enable-sockets. Required for user script support.
PHP multibyte support		php-mbstring, --enable-mbstring
IBM DB2 **ibm_db2**		Required if IBM DB2 is used as Zabbix back end database.
MySQL **php-mysql**	3.22 or later	Required if MySQL is used as Zabbix back end database.
Oracle **oci8**		Required if Oracle is used as Zabbix back-end database.
PostgreSQL **php-pgsql**	7.0.2 or later if Zabbix < 1.8.9 7.4 or later if Zabbix >= 1.8.9	Required if PostgreSQL is used as Zabbix back-end database. Consider using PostgreSQL 8.x or later for much better performance. It is suggested to use at least PostgreSQL 8.3, which introduced much better VACUUM performance.
SQLite **php-sqlite3**	3.3.5 or later	Required if SQLite is used as Zabbix back-end database.

Figura 4.17: Zabbix: Tabela ilustrativa com os requisitos de software necessários. Extraído de (ZABBIX, 2012)

- Mac OS X.

É possível perceber que não existe a possibilidade de instalar o servidor em máquinas com o sistema operacional Microsoft Windows em nenhuma das ferramentas. Entretanto, em todas elas é possível monitorar uma máquina Windows remotamente.

Quanto à compatibilidade de softwares necessários para rodar os servidores, o Zabbix foi a ferramenta que forneceu mais detalhes de suas necessidades, que podem ser observados na Figura 4.17.

O Nagios é uma ferramenta que não exige muitos aplicativos para serem instalados previamente. Junto com o pacote de instalação do Zenoss também serão instalados todos os outros softwares necessários para o mesmo rodar (mesmo caso do Nagios).

4.4.2 Alertas

Uma outra funcionalidade bastante importante é a capacidade de gerar alertas. Em algumas empresas, existem times focados à monitoração que cuidam dos alertas gerados (FERNANDES, 2011). Para esses times, é de extrema importância que a ferramenta de

Tabela 4.7: Tipos de Alertas Possíveis

	Zabbix	Nagios	Zenoss
Email	Sim	Sim	Sim
Mensagem de Texto (SMS)	Sim	Sim	Sim
Ligação Telefônica	Não	Sim	Sim
Script	Sim	Sim	Não

monitoração possua uma maneira de comunicar os alertas que acontecem.

Um alerta deve ser gerado sempre que um monitor estiver apontando um problema. O monitor pode ser configurado para alertar quando ultrapassar um valor limiar fixo, por exemplo quando o uso de CPU ultrapassar 95%.

A Tabela 4.7 mostra as possibilidades de alertas que são possíveis de serem gerados a partir das ferramentas. A possibilidade de executar um script remotamente é, sem dúvida nenhuma, a mais interessante pois poderia permitir que o script arrumasse o problema sem que fosse necessária uma intervenção humana.

Um exemplo de utilização de um script para resolver um problema pode ser dado da seguinte maneira: suponha que existe um monitor que cuida para que um determinado serviço esteja sempre rodando. Se o monitor detectar que esse serviço não está mais executando, um script poderia ser rodado para iniciar esse serviço, não sendo necessária intervenção humana para iniciar o serviço.

É possível de se observar que o Nagios é a ferramenta mais completa, suportando todas as operações. Com o Zabbix não é possível realizar ligações telefônicas (mensagens gravadas) e com o Zenoss não é possível a execução de scripts.

4.4.3 Escalabilidade

Em algumas empresas, existem centenas, as vezes milhares de computadores conectados em uma mesma rede. É muito importante que essa imensa quantidade de computadores seja monitorada. Para isso ser possível, as ferramentas de monitoramento utilizadas devem ser altamente escaláveis.

Nos próximos parágrafos, será feita uma investigação sobre a capacidade de nodos (servidores) em cada uma das ferramentas. Também será tentado imaginar como o aumento significável do número de servidores (de 3 servidores para 1000, por exemplo) impactaria na interface e visualização da ferramenta. Essa análise será hipotética.

Todas as ferramentas apresentaram uma boa documentação quanto aos requisitos mínimos de hardware que devem existir a medida que a rede de servidores monitorados for crescendo. As tabelas podem ser encontradas na Figura 4.18.

Pode ser observado que o Zenoss é, claramente, a ferramenta que mais consome os recursos da máquina. Ela é a que mais necessita espaço em disco e memória RAM para conseguir rodar satisfatoriamente.

Quanto ao impacto na interface das ferramentas, isso não parece ser um problema para o Zabbix e o Zenoss. Elas são bastante estruturadas e possuem áreas distintas para monitores, servidores e problemas. Já no Nagios, ao utilizar a interface padrão, isso poderia ser um problema pois todos os objetos ficam listados no mesmo lugar, o que tonaria a visualização muito poluída.

Figura 4.18: Requisitos mínimos de hardware

Name	Platform	CPU/Memory	Database	Monitored hosts
Small	Ubuntu Linux	PII 350MHz 256MB	MySQL MyISAM	20
Medium	Ubuntu Linux 64 bit	AMD Athlon 3200+ 2GB	MySQL InnoDB	500
Large	Ubuntu Linux 64 bit	Intel Dual Core 6400 4GB	RAID10 MySQL InnoDB or PostgreSQL	>1000
Very large	RedHat Enterprise	Intel Xeon 2xCPU 8GB	Fast RAID10 MySQL InnoDB or PostgreSQL	>10000

(a) Zabbix. Extraído de (ZABBIX, 2012)

Monitored Nodes / Hosts	Monitored Services	Hard Drive Space	CPU Cores	RAM
50	250	40 GB	1 – 2	1 – 4 GB
100	500	80 GB	2 – 4	4 – 8 GB
> 500	> 2500	120 GB	> 4	> 8 GB

(b) Nagios. Extraído de (NAGIOS, 2012)

Deployment Size	Memory	CPU	Storage
1 to 250 devices	4GB	2 cores	1x300GB, 10K RPM drive
250 to 500 devices	8GB	4 cores	1x300GB, 10K RPM drive
500 to 1000 devices	16GB	8 cores	1x300GB, 15K RPM drive
1000 to 2000 devices	64GB	8 cores	1x300GB, 15K RPM drive

(c) Zenoss. Extraído de (ZENOSS, 2012)

4.4.4 Configuração e Personalização

Nesta parte do trabalho, será avaliada a capacidade de adaptabilidade e personalização das ferramentas. As maneiras que o usuário tem de configurar a interface do programa (sem ter de modificar o código) para fazer com que a ferramenta se adeque às suas necessidades.

A interface do Nagios não é facilmente manipulada. Ela possui várias restrições de uso e foi a que menos mostrou adaptabilidade nesse quesito.

Já as interfaces do Zabbix e do Zenoss se mostraram bastante fáceis de serem reconfiguradas. Em ambas as ferramentas é possível re-ordenar os elementos gráficos. Também é possível adicionar e remover os elementos disponíveis em outras partes do programa na página inicial, o que acaba facilitando a navegação.

A configuração da interface do Zenoss se mostrou um pouco melhor de ser configurada do que a do Zabbix pois são disponibilizadas mais opções para o usuário.

4.5 Tabela Final

Nesse capítulo, foram mostrados diversos atributos e critérios para avaliar as ferramentas. Foram obtidos muitos resultados diferentes em cada uma das análises. Agora será feita uma tabela comparativa trazendo todos os resultados.

De maneira a simplificar a visualização dos resultados, será feito um ranking da ferramenta com melhor performance em cada um dos resultados, sendo que cada ferramenta será dada uma nota de 1 a 3, sendo 1 a pior ferramenta para aquele quesito e 3 a melhor ferramenta. A Tabela 4.8 traz o resultado da comparação final entre as ferramentas e todos os critérios avaliados.

Observando a Tabela 4.9 é possível identificar que a ferramenta Zabbix se mostrou a mais completa. Ela é um misto de interface gráfica boa e muitas funcionalidades interessantes.

Tabela 4.8: Tabela final de comparação

	Zabbix	Nagios	Zenoss
Documentação e Disponibilidade	3	2	1
Código Fonte	2	3	1
Fóruns	1	3	2
Suporte	3	1	2
Instalação	2	3	1
Interface com Usuário	3	1	2
Criação de Monitores	3	2	1
Estado dos Monitores	3	2	1
Gráficos de Análise	3	1	2
Relatórios	2	1	3
Usabilidade Geral	1	2	2
Capacidade	1	3	2
Compatibilidade	1	2	2
Alertas	2	3	1
Escalabilidade	3	1	2
Configuração e Personalização	2	1	3

Tabela 4.9: Tabela final de comparação com análises

	Zabbix	Nagios	Zenoss
Média	2,1875	1,9375	1,75
Vezes Melhor	7	5	2
Vezes Pior	4	6	6

Em segundo lugar na avaliação fica a ferramenta Nagios. Ela mostrou ser uma ferramenta muito adaptável, apesar de exigir um usuário mais avançado que não precise utilizar tanto a interface gráfica.

O Zenoss ficou na última colocação. Apesar de sua interface gráfica extremamente poderosa, essa ferramenta se mostrou inferior às outras justamente por causa da dificuldade de uso. Ela é bastante grande, incluindo até os plugins do Nagios, mas que precisa ter sua manipulação de informações melhorada.

No próximo capítulo será feita uma análise de um estudo de caso. Será feita uma abordagem prática ao instalar e configurar as ferramentas e monitores. Serão medidos fatores para ter certeza que as ferramentas atendem os requisitos mínimos de funcionalidade.

5 ESTUDO DE CASO

Para comparar a eficiência e a praticidade dos softwares apresentados, estudos de caso foram elaborados a partir de um objetivo inicial: monitorar um determinado serviço. Foram analisadas as principais propriedades de cada um deles fazendo testes que forçaram que os monitores falhassem.

Todas as ferramentas foram instaladas em máquinas virtuais criadas utilizando o software gerenciador de máquinas virtuais Oracle VM Virtual Box. Esse programa foi instalado em um notebook com a seguinte configuração:

- Processador: Intel Core i7-2640M 2,80 GHz;
- Memória RAM: 6 GB;
- Memória Disco: 685 GB;
- Sistema Operacional: Windows 7 Professional.

5.1 Configurações Básicas

As simulações ocorreram da seguinte forma: uma máquina virtual foi configurada com uma das ferramentas de monitoração e outra foi configurada para ser o servidor monitorado. Foram configuradas 6 máquinas virtuais (duas para cada ferramenta). Cada máquina virtual teve a seguinte configuração:

- Processador: Intel Core i7-2640M 2,80 GHz;
- Memória RAM: 2 GB;
- Memória Disco: 20 GB;
- Sistema Operacional: Ubuntu 12.04.

As ferramentas instaladas estão na seguinte versão:

- Zabbix 2.0.3;
- Nagios 3.2.3;
- Zenoss 4.2.0.

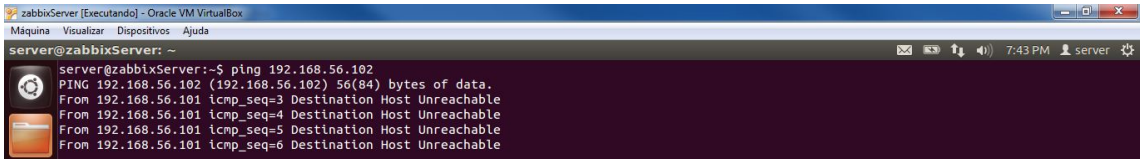


Figura 5.1: Ping para um servidor desligado

5.2 Principais Indicadores

Nas subseções a seguir, serão descritos alguns dos principais indicadores de que um servidor está funcional. O objetivo deste trabalho não é analisar nenhuma aplicação específica. Por isso, os indicadores escolhidos são os mais gerais possíveis.

Para cada um dos fatores descritos abaixo, foi configurado um monitor nas 3 ferramentas escolhidas. Foram simuladas situações diversas em que esses monitores deveriam detectar um problema para se confirmar a eficiência da monitoração.

5.2.1 Ping

O Ping é um utilitário de administração de redes de computadores utilizado para testar se um servidor é alcançável por outro através da Internet. O Ping opera mandando uma mensagem que utiliza o protocolo ICMP (Internet Control Message Protocol) para um determinado servidor e analisa e resposta (ou a falta dela) (HALABI, 1997).

O protocolo ICMP foi criado com o intuito de padronizar a confirmação (ou negação) de que um mensagem foi recebida (SILVA CARISSIMI; ROCHOL; GRANVILLE, 2009). Dessa maneira, o servidor que possui a ferramenta de monitoração instalado estará, através do comando de Ping, validando se a outra máquina está acessível.

Se um problema desse tipo acontecer, significa que o servidor cliente, que está com a aplicação monitorada rodando, não está acessível através da Internet. Ou seja, a máquina está desligada, trancada ou com problemas de rede. Esse é um problema muito grave pois inutiliza totalmente a máquina.

Um exemplo de problema de Ping pode ser observado na Figura 5.1. Nessa Figura, um servidor está tentando validar deu caminho até outra máquina mas não consegue porque ela está desligada.

No contexto deste trabalho, para testar a monitoração sobre o Ping de uma máquina virtual basta desligar a máquina virtual (se fosse em um ambiente real, isso significaria desligar a máquina).

5.2.2 Uso de CPU

Um monitor que cuida do uso de CPU de um computador é extremamente importante. Um monitor desse tipo cuida para que o processador da máquina não seja usado por muito tempo perto do seu limite.

Um computador que fica muito tempo rodando com o uso do processador em 100% tem grandes chances de começar a apresentar falhas, uma vez que as operações começarão a acumular e podem não ser processadas corretamente.

Monitores desse tipo funcionam com valor setado de limiar. O monitor alertará sempre que o percentual de utilização do processador ultrapassar esse limiar. Esse percentual pode ser medido estaticamente (uma única amostra) ou pode ser medido como uma média dos últimos minutos (mais de uma amostra).

Um exemplo de máquina com alto uso do processador pode ser encontrado na Figura

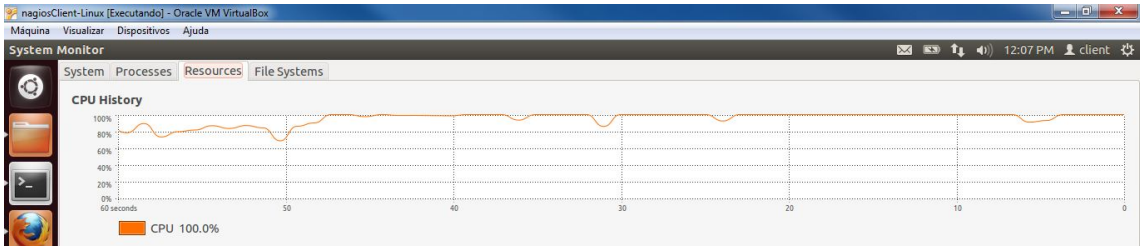


Figura 5.2: Alto uso de CPU

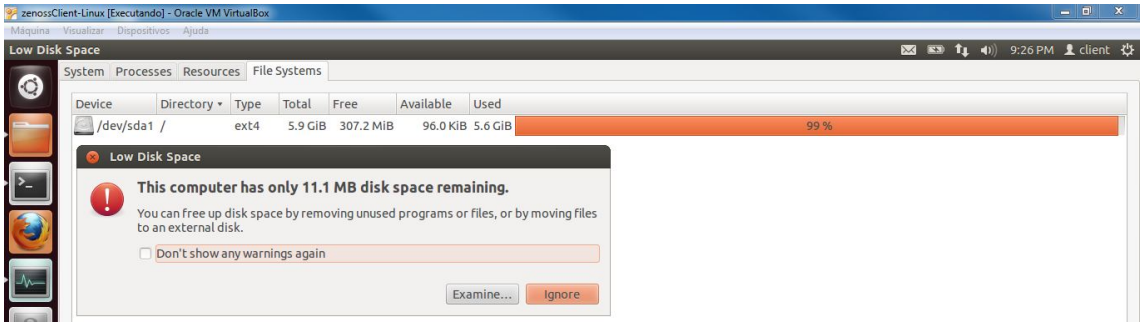


Figura 5.3: Pouco espaço livre em Disco

5.2. Nela, podemos observar que o uso do processador chegou a 100%. Por causa disso, a máquina fica extremamente lenta e muito difícil de ser usada.

Entretanto, a maneira mais prática de testar a corretude de um monitor de uso de CPU é diminuindo o limite de utilização. Por exemplo, se o monitor está configurado para alertar sempre que o uso de CPU estiver acima de 95%, basta baixar esse valor para 0%. Depois que o monitor for validado, o valor limite pode ser restaurado ao valor inicial.

5.2.3 Espaço Livre em Disco

O avanço tecnológico e o constante progresso do desenvolvimento de processamento de dados vem resultando num crescente volume de dados a serem gerenciados (SILVEIRA, 2012). Fica cada vez mais importante monitorar a quantidade de memória disponível nas máquinas.

Muitas tecnologias comerciais salvam muitos arquivos de logs de execução. Esses arquivos podem chegar a ocupar centenas de Mega Bytes por dia (SCHAEFER et al., 2008). Com isso, vemos um grande problema que é a gerência e controle desses arquivos.

Se faltar espaço no disco no cliente para salvar algum dado importante para uma aplicação específica, pode ser que essa aplicação comece a não mais responder até que haja espaço em livre para salvar os logs ou arquivos (KAMIN, 1988).

Assim, monitores com essa funcionalidade são configurados para alertar caso o percentual de espaço em disco livre fique menor que um determinado valo limítrofe. O objetivo é que o administrador do servidor tenha tempo de remover arquivos que não são mais necessários.

Um exemplo de problema pode ser encontrado na Figura 5.3. Aparentemente isso não gera nenhum problema imediato na máquina. No entanto, a medida que as aplicações começam a rodar e salvar seus logs e arquivos, problemas começariam a ser vistos por não haver onde salvá-los. Os próprios arquivos de log do sistema operacional (Windows, Linux) passam a ser problema nesses casos.

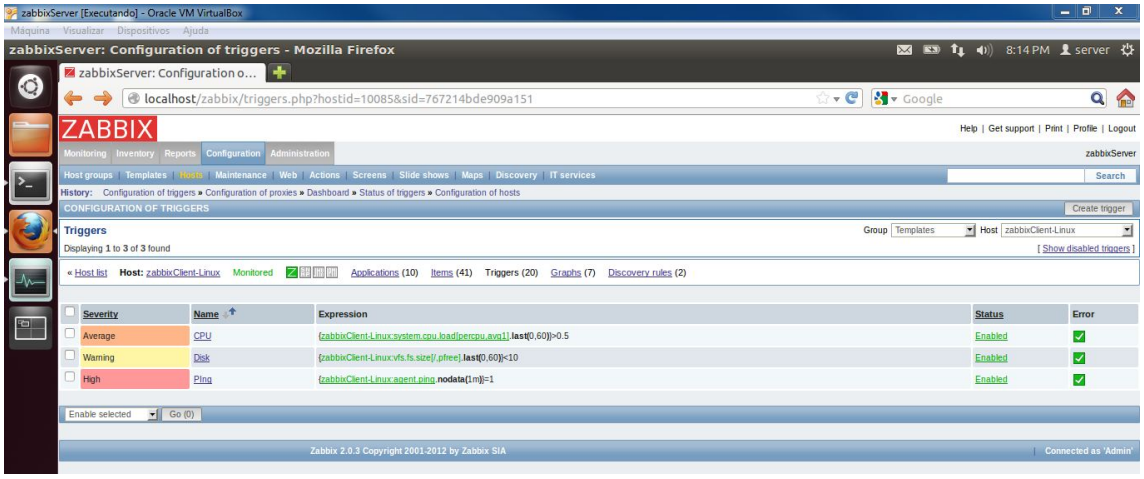


Figura 5.4: Zabbix: Interface após configuração dos monitores

Para confirmar que esse monitor está configurado corretamente é possível mudar do mesmo modo que no item anterior: temporariamente diminuir o limiar para 0%.

5.3 Aplicação em Servidor Linux

Nos parágrafos abaixo, será descrito como cada um dos indicadores descritos acima foram implementados nas ferramentas. Será possível observar as diferentes possibilidades de monitoração existentes visto que cada um dos programas utiliza um método diferente.

Após a configuração dos monitores, o problema será simulado segundo indicado anteriormente (para cada um dos indicadores existe uma maneira diferente de simular o problema). Será analisado o desempenho das ferramentas durante a detecção das falhas, bem como a visualização das mesmas na interface e os alertas gerados.

5.3.1 Zabbix

A criação de monitores no Zabbix é feita através da interface. Para criar cada um dos monitores acima basta selecionar um dos modelos pré-configurados no sistema, escolher qual a frequência do monitor e o valor limite.

Após a criação dos três monitores, chega-se à tela que pode ser observada na Figura 5.4. Nela podemos observar o estado de cada um dos novos monitores criados, bem como a severidade deles e a indicação de eles estarem alertando ou não.

Seguindo os modelos descritos previamente para simular uma situação de falha, foi possível observar que o Zabbix alertou dentro do tempo esperado (frequência definida para cada alerta).

A visualização dos monitores em estado de erro pode ser observada na tela inicial do Zabbix, mostrada na Figura 5.5. Como pode ser observado, já nesta tela inicial é possível obter informações importantes como a quanto tempo o problema está acontecendo e qual o servidor com problemas.

Uma funcionalidade que facilita a investigação do usuário é a possibilidade de, utilizando a interface, descobrir o estado do monitor a qualquer momento. Isso faz com que a investigação seja mais rápida.

Durante a execução dos testes a máquina virtual não apresentou significativas mudanças de performance, permanecendo com uso de CPU constante em 35% e uso de memória

Zabbix

Figura 5.5: Zabbix: Página inicial em que é possível localizar o monitor falhado

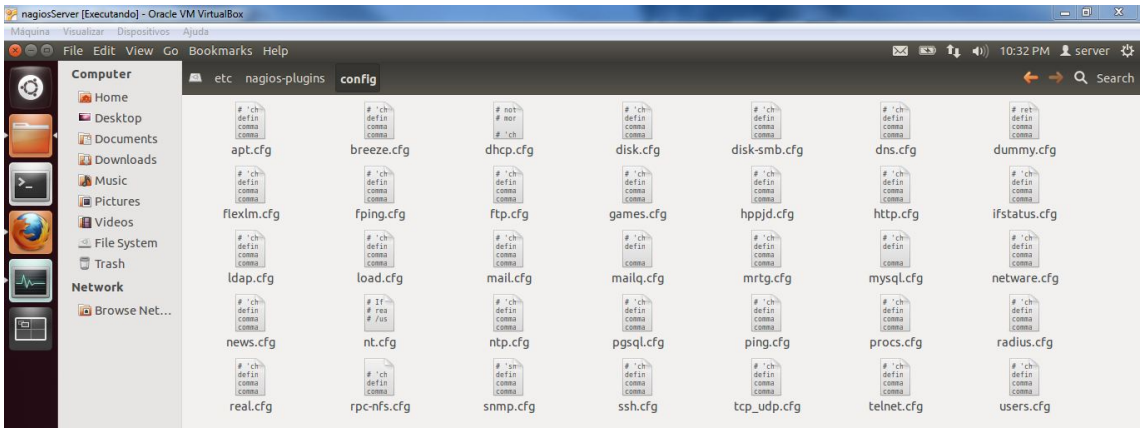


Figura 5.6: Nagios: Plugins instalados na versão padrão

em 23%.

5.3.2 Nagios

Para criar os monitores na ferramenta Nagios, é necessário modificar os arquivos de configuração do mesmo. No pacote padrão de instalação desse programa, não existe uma interface que facilite a configuração de novos monitores. Tudo deve ser feito utilizando os arquivos de configuração. Apesar de não ter uma interface de configuração amigável, o Nagios é fácil de ser configurado uma vez que se sabe onde todos os arquivos estão e quais devem ser modificados.

Todos os comandos executados pelo Nagios são, na verdade, linhas de comando que chamam plugins. Muitos plugins são instalados junto com o pacote padrão. Todos os monitores implementados para esse trabalho foram encontrados nos pacotes padrões. A Figura 5.6 mostra alguns desses pacotes.

Para configurar os monitores, devem ser criados serviços que irão chamar os plugins. A declaração e sintaxe desses serviços deve seguir os padrões estabelecidos pelo Nagios. A Figura 5.7 mostra a configuração do serviço de Ping que foi configurado.

Estando todos os novos monitores prontos nos arquivos de configuração, ainda é pre-

```
#ping service
define service{
        host_name               nagiosClient-Linux
        service_description     Ping
        check_command           check_ping!100.0,20%!500.0,60%
        max_check_attempts      3
        check_interval          1
        retry_interval          1
        check_period            24x7
        notification_interval   30
        notification_period     24x7
        notification_options    w,c,r
        contact_groups
}
```

Figura 5.7: Nagios: Exemplo de configuração de um serviço

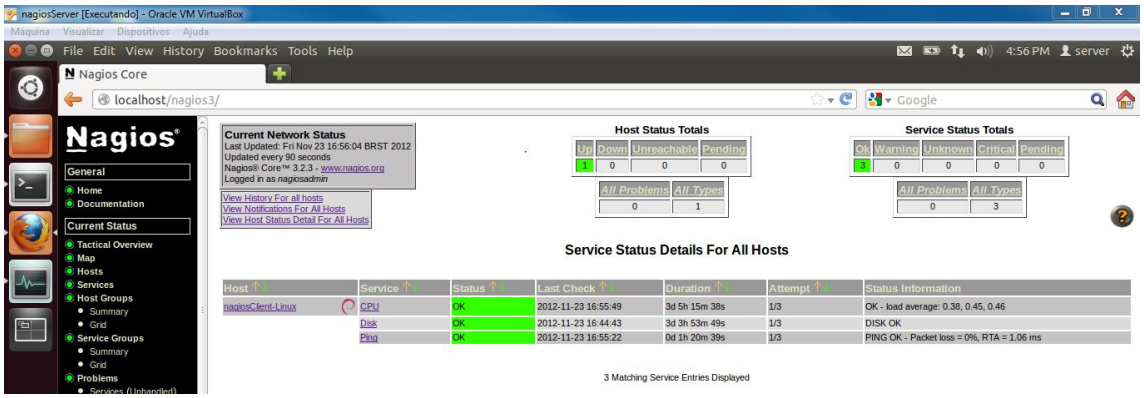


Figura 5.8: Nagios: Interface após configuração dos monitores

ciso reiniciar o serviço do Nagios para que as mudanças tenham efeito. Nota-se aqui que as mudanças custam um pouco mais de tempo para serem efetuadas e refletir no sistema.

Após a criação dos três serviços, chega-se a uma tela que possui dados de todos os monitores criados. Essa página pode ser visualizada na Figura 5.8. Da mesma maneira que com a ferramenta Zabbix, a monitoração também pode ser feita a partir dessa tela, pois temos os detalhes do estado atual de cada um dos monitores.

Utilizando um dos métodos para simular uma situação de falha, pode-se observar que essa ferramenta teve um tempo de atualização um pouco superior ao esperado. O intervalo em que a página atualiza é um valor fixo pré-determinado que não pode ser modificado e acabou sendo maior do que a frequência dos monitores criados. Entretanto, alertas foram disparados no momento correto. A Figura 5.9 permite a visualização da chamada "Visão Tática da Monitoração"do Nagios. Nessa página podemos visualizar o estado de todos os servidores configurados bem como todos os serviços.

5.3.3 Zenoss

Toda os monitores do Zenoss são configurados, assim como no Zabbix, pela interface gráfica. A primeira parte para se configurar um monitor é cadastrar o dispositivo (servidor) na ferramenta. Essa ação traz uma grande vantagem para o usuário pois, ao fazer isso, o Zenoss irá automaticamente modelar a máquina alvo utilizando seus plugins. Com isso, será possível obter diversas informações e gráficos importantes sobre cada dispositivo.

A criação dos monitores deve ser feita utilizando alguma das classes de evento existentes (uma classe representa um plugin). Algumas das classes existentes podem ser visualizadas na Figura 5.10. A partir da seleção de uma classe, é possível escolher uma sub-classe dessa classe, havendo a disponibilidade. A hierarquia de classes e sub-classes pode ser tão grande quando se queira.

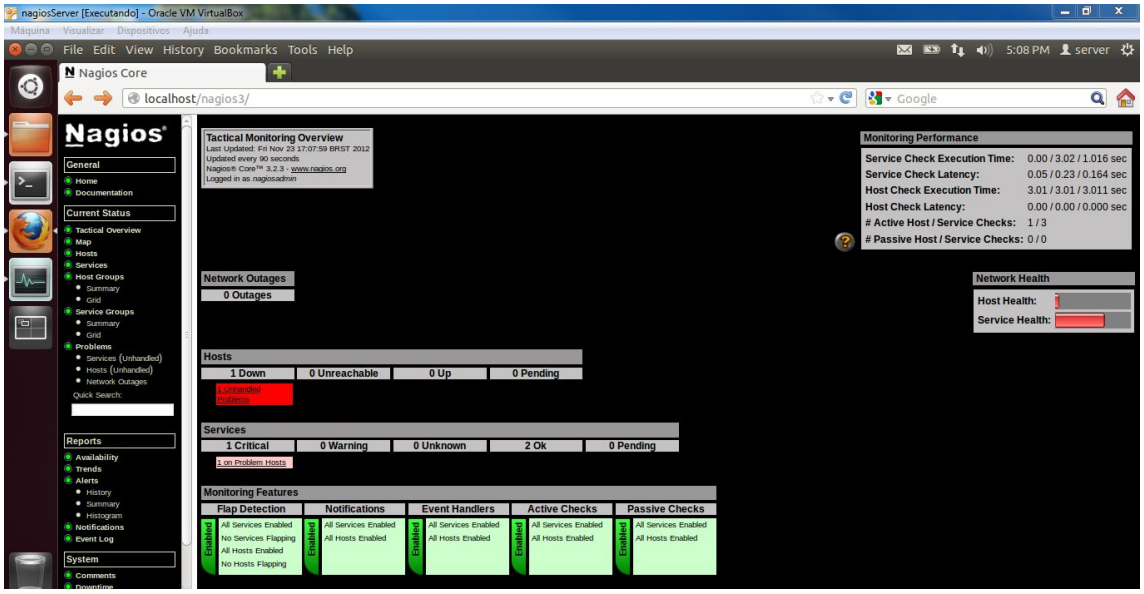


Figura 5.9: Nagios: Visão Tática mostrando um monitor alertando

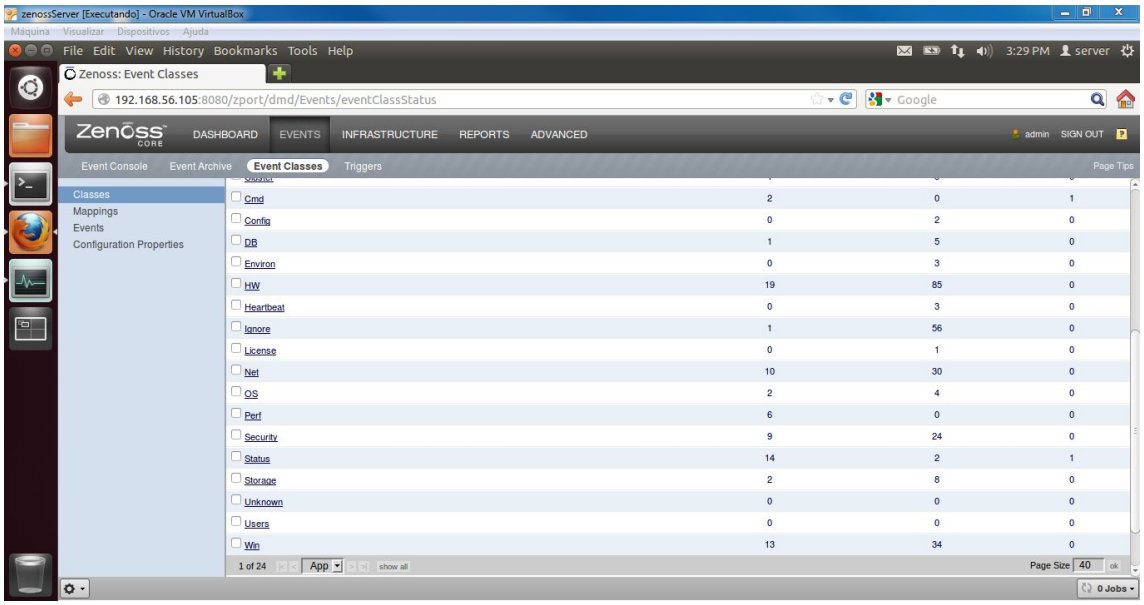


Figura 5.10: Zenoss: Algumas das classes de eventos existentes

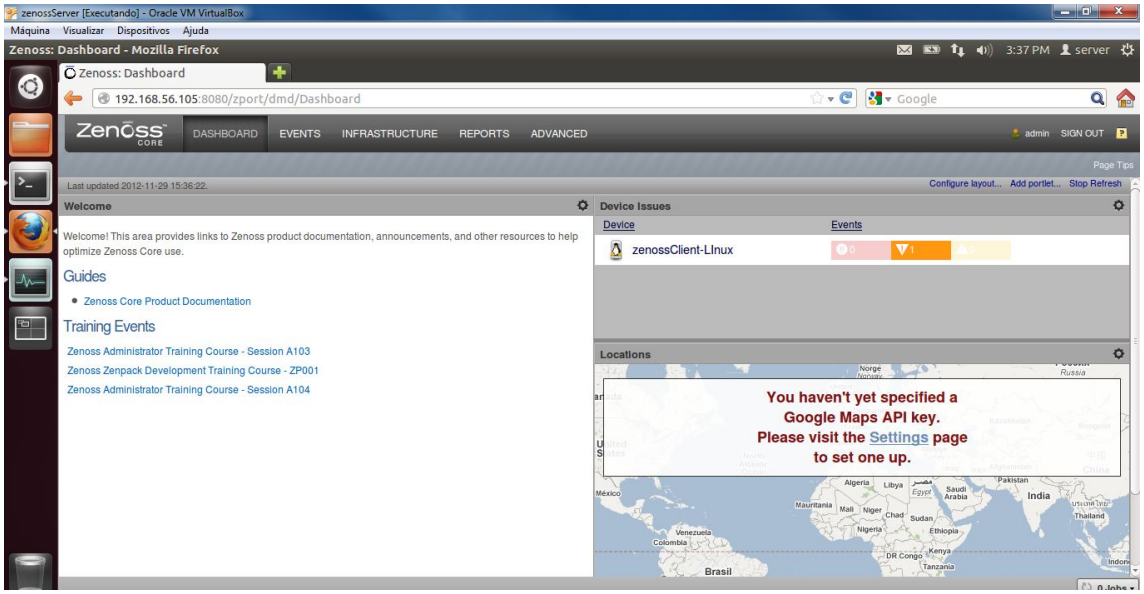


Figura 5.11: Zenoss: Tela inicial

Figura 5.12: Zenoss: Alto uso de CPU e memória RAM ao iniciar

Após a configuração e criação dos eventos, foi possível observar que faltou um pouco de comunicação do estado das solicitações. A única maneira encontrada de se certificar que os eventos realmente haviam sido criados com sucesso foi partir para a fase de testes.

A tela inicial do Zenoss, observada na Figura 5.11, pode ser configurada para mostrar a localização física das máquinas (utilizando plugin do Google Maps). Nessa Figura é possível observar a visualização do usuário quando existe um monitor em estado de falha.

Neste estudo, foi possível observar que o Zenoss alertou sobre a falha do monitor dentro do tempo esperado, não apresentando nenhum problema a partir dessa parte. Foi possível retornar o monitor ao estado normal sem grande esforço.

Um fato importante observado neste experimento foi a grande quantidade de CPU e memória RAM utilizados pelo Zenoss no servidor. A Figura 5.12 mostra o gradual aumento do uso de memória quando o Zenoss começa a ser utilizado na máquina que está efetuando a monitoração. Esse comportamento não foi observado nas outras ferramentas.

5.4 Resultado da Comparação

De todas as ferramentas, a que mostrou o melhor desempenho durante todos os testes foi a ferramenta Zabbix. Mais uma vez ela mostrou ser bastante completa e precisa. A configuração dos monitores foi bastante intuitiva, sempre utilizando a interface gráfica do programa.

O software Nagios também teve um desempenho bastante satisfatório. Apesar de a sua configuração não ser tão simples e fácil quanto à do Zabbix, para usuários acostumados a usá-la ela mostrou ser eficiente. Foi a ferramenta que menos utilizou recursos da máquina em que estava instalada, devido à pouca utilização da interface gráfica.

O Zenoss mostrou, mais uma vez, ter um potencial bastante bom. É possível utilizá-lo em conjunto com o Google Maps e diversos outros plugins para tornar a experiência do usuário a mais agradável possível. Entretanto, talvez ao tentar ser muito completa, a interface acabou ficando difícil de entender e nada simples de ser utilizada. O alto consumo dos recursos da máquina servidor também pontuam contra essa ferramenta.

6 CONCLUSÃO

Esse estudo mostrou a importância de uma empresa que têm seus serviços voltados ao cliente (por exemplo, uma empresa de jogos online) possuir uma ferramenta de monitoramento completa. É importante que ela possa se comunicar facilmente com seus usuários e ainda ajudá-los nas análises que eles têm de fazer.

Mais que isso, um bom software de monitoria deve ser completo e capaz de realizar o tipo de monitoração exigida pelas aplicações das empresas. Em um ambiente estabelecido, a monitoração deve se adaptar a ele para obter os melhores resultados.

6.1 Resultados Finais

Durante esse estudo, foi possível observar claramente o foco que cada uma das ferramentas recebeu em seu desenvolvimento. O Nagios foi uma ferramenta que mostrou ser facilmente adaptável e melhorada, por sua facilidade de receber novos plugins. Inclusive o desenvolvimento desses plugins é bastante facilitado por uma comunidade ativa bastante grande.

A ferramenta Zenoss foi a que mostrou o pior desempenho entre as três. Ela se mostrou extremamente complicado de instalar e configurar, com uma interface muito bonita mas com baixa usabilidade. O Zabbix, pelo contrário, foi a ferramenta mais completa. Possui uma interface gráfica muito boa e também é capaz de ser utilizada de muitas maneiras diferentes.

6.2 Trabalhos Futuros

Uma possibilidade de aumentar o escopo deste trabalho é a inclusão e comparação com ferramentas vendidas comercialmente. Medir suas performances e comportamentos seria bastante interessante de ser observado para saber se seria melhor apostar em uma ferramenta livre ou em uma paga.

Outra oportunidade de melhoria deste trabalho seria um estudo mais detalhado sobre os tipos de monitores disponíveis de serem configurados. A possibilidade de existir monitores que sigam uma série de passos em uma página da internet, por exemplo, é uma habilidade bastante interessante para uma ferramenta de monitoração e sem dúvida a deixaria mais completa.

REFERÊNCIAS

BALIS, B.; BUBAK, M.; FUNIKA, W.; SZEPIENIEC, T.; WISMüLLER, R. An Infrastructure for Grid Application Monitoring. In: **Recent Advances in Parallel Virtual Machine and Message Passing Interface**. [S.l.]: Springer Berlin Heidelberg, 2002.

CASE, J.; FEDOR, M.; SCHOFFSTALL, M.; DAVIN, J. **Simple Network Management Protocol (SNMP)**. [S.l.: s.n.], 1990.

COLLECTIVE, C. E. **http**://cec.vcn.bc.ca/cmp/modules/mon-wht.htm. 2012.

DUBIE, D. Cheap Tools to Try During Tough Times. **PC World**, [S.l.], 2009.

FEIT, S. M. **SNMP**: a guide to network management. [S.l.]: McGraw-Hill Professional, 1993.

FERNANDES, L. **A Importância da Monitoração de Desempenho das Aplicações Web com Foco no Negócio**. 2011.

HALABI, B. **Internet Routing Architectures**. [S.l.]: Cisco Systems, 1997.

IMAMAGIC, E.; DOBRENIC, D. Grid infrastructure monitoring system based on Nagios. In: GRID MONITORING, 2007., 2007. **Proceedings...** [S.l.: s.n.], 2007.

INC, Z. **Zenoss Administration 2.3**. [S.l.: s.n.], 2009.

INC, Z. **Zenoss Core Administration 4.2**. [S.l.: s.n.], 2012.

CIENTíFICOS EDITORA, L. T. e (Ed.). **Disco Rígido no PC**. [S.l.]: Sybex, 1988.

KOCH, M. **Uma Proposta de Solução de Gerenciamento de Contabilização utilizando Nagios e Cacti**. 2008.

Linux World Magazine. [S.l.]: SYS CON, 2006. v.4, n.4.

MSDN. **http**://msdn.microsoft.com. 2012.

NAGIOS. **http**://www.nagios.org/. 2012.

NETO, F. H. G. **Guardião**: uma ferramenta para monitoração de serviços e servidores de rede. 2001. Dissertação (Mestrado em Ciência da Computação) — Universidade Federal do Rio Grande do Sul.

OLUPS, R. **Zabbix 1.8 Network Monitoring**. [S.l.: s.n.], 2010.

PERVILä, M. Using Nagios to monitor faults in a self-healing environment. **Seminar on Self-Healing Systems, University of Helsinki**, [S.l.], 2007.

PHAAL, P. **Network Monitoring Device and System**. [S.l.: s.n.], 1994.

POSTEL, J.; REYNOLDS, J. **Telnet Protocol Specification**. RFC 854.ed. [S.l.]: ISI, 1983.

RAKOSHITZ, G.; VAID, A.; PANDIT, A.; PUTTA, S. **Traffic Monitoring Tool for Bandwidth Management**. [S.l.: s.n.], 2003.

SCHAEFER, K.; COCHRAN, J.; FORSYTH, S.; BAUGH, R.; EVEREST, M.; GLENDENNING, D. **Professional IIS 7**. [S.l.]: wrox, 2008.

SILVA CARISSIMI, A. da; ROCHOL, J.; GRANVILLE, L. Z. **Redes de Computadores**. [S.l.]: Bookman, 2009.

SILVEIRA, R. P. **Novas Metodologias para Compressão de Dados de Processos e para o Ajuste do Sistema PI**. 2012. Dissertação (Mestrado em Ciência da Computação) — Universidade Federal do Rio Grande do SUl.

TURNBULL, J. **Pro Nagios 2.0**. [S.l.: s.n.], 2006.

URLOCKER, Z. How open source gains market share. **Info World**, [S.l.], 2009.

YLONEN, Y.; LONVICK, C. **The Secure Shell (SSH) Protocol Architecture**. RFC 4251.ed. [S.l.]: Cisco Systems, Inc, 2006.

ZABBIX. **http**://www.zabbix.com/. 2012.

ZENOSS. **http**://www.zenoss.com/. 2012.