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

MÁRCIO BARBOSA DE CARVALHO

Adaptação da Ferramenta Nagios para o Monitoramento de Servidores Virtuais

Trabalho de Graduação.

Prof. Dr. Lisandro Zambenedetti Granville
Orientador

Porto Alegre, junho de 2010.

SUMÁRIO

LISTA DE ABREVIATURAS E SIGLAS

CPU	Unidade Central de Processamento
VMM	Virtual Machine Monitor
NRPE	Nagios Remote Plug-in Executor
NSCA	Nagios Service Check Acceptor
NEB	Nagios Event Broker
SSH	Secure Shell
ePN	Embedded Perl Nagios
MB	Megabyte
GB	Gigabyte
TB	Terabyte
AVA	Ambiente Virtual de Aprendizado
DC	Domain Controller

LISTA DE FIGURAS

LISTA DE TABELAS

RESUMO

A grande complexidade dos ambientes computacionais atuais torna as tarefas dos administradores também complexas. Esta complexidade requer que o administrador utilize muitas ferramentas para conseguir visualizar seu ambiente. Com o uso de virtualização, este cenário torna-se mais complexo. A complexidade das tarefas dos administradores também aumenta, já que cada ambiente de virtualização requer uma nova ferramenta específica para este ambiente.

Este trabalho propõe uma solução que integra as informações de monitoramento de ambientes virtualizados, disponíveis em ferramentas específicas para determinadas plataformas, em uma ferramenta de monitoramento popular. A ferramenta de monitoramento escolhida foi o Nagios, porque é uma ferramenta de código aberto e amplamente empregada em organizações. Ao decorrer do trabalho, serão apresentados os componentes e estratégias para tornar isto possível. Uma avaliação da solução proposta será feita em um ambiente real. Os resultados da avaliação serão apresentados em comparativos de tempo e utilização de rede para as estratégias apresentadas. Além disso, um teste de escalabilidade será executado para mostrar o comportamento da solução proposta em sistemas de grande porte.

Palavras-Chave: monitoramento adaptativo, monitoramento de máquinas virtuais.

Adapting the Nagios tool for monitoring virtual servers

ABSTRACT

The great complexity of current computing environments makes the tasks of administrators also more complex. This requires that the administrator use several tools to visualize your environment. With the use of virtualization this scenario becomes more complex. The complexity of the tasks of the administrators also increases because each virtualization environment requires a new tool specifically for them.

This work proposes a solution that integrates monitoring information of virtualized environments, available in platform specifics tools, into a popular monitoring tool. The monitoring tool chosen is Nagios because is an open source tool and is widely employed in enterprises. Along this work we present the components and strategies to make it possible. An evaluation of the proposed solution is made in a real environment. The evaluation results are presented with comparisons of time and network usage for these strategies. Moreover, a scalability test is made to show the behavior of the proposed solution in large systems.

Keywords: adaptative monitoring, virtual machine monitoring.

1 INTRODUÇÃO

Soluções modernas para virtualização de servidores como Citrix XenServer e VMware vSphere, apesar de permitirem aos administradores de rede a criação de infra-estruturas de tecnologia da informação mais robustas e flexíveis, quebram os processos de monitoramento empregados por sistemas de monitoramento populares como Nagios e ZABBIX. Isto porque sistemas como esses possuem uma lista fixa de servidores para contatar quando monitoram os serviços de um ambiente de rede, qualquer migração de um servidor virtual ou substituição de um hardware defeituoso invalida a lógica de monitoramento imediatamente. Deve-se notar que estas soluções para virtualização são naturalmente dinâmicas tomando a iniciativa das migrações para balanceamento de carga e tolerância às falhas.

Com as ferramentas de monitoramento atuais, os administradores de rede devem ajustar manualmente esta ferramenta para refletir a situação atual do ambiente. Entretanto, a quantidade de equipamentos e serviços sob a sua responsabilidade não permite que uma estratégia deste tipo seja adotada. Assim, adaptações nas ferramentas de monitoramento são necessárias para tratar a dinâmica natural dos novos ambientes computacionais.

O restante do trabalho é organizado da seguinte forma: no capítulo 2, são apresentados fundamentos sobre virtualização e sobre o monitoramento de ambientes computacionais. Neste capítulo, também serão apresentadas características e componentes do Citrix XenServer e da ferramenta de monitoramento Nagios, que são empregados no ambiente monitorado. No capítulo 3, o problema descrito acima sucintamente será definido em detalhes e serão apresentados os requisitos funcionais e qualitativos para uma solução. Neste capítulo também será proposta uma solução para este problema. Fazem parte desta solução seus componentes empregados, bem como as diferentes estratégias apresentadas utilizando estes componentes. No capítulo 4, são apresentados detalhes da implementação destes componentes. No capítulo 5, são feitas avaliações em um ambiente real para verificar o funcionamento da solução proposta. Essas avaliações levam em consideração as diferentes estratégias apresentadas, traçando um comparativo entre elas quanto às estatísticas de tempo e de utilização de rede. Neste capítulo, também será feito um teste de escalabilidade para verificar o funcionamento da solução quando o ambiente monitorado emprega centenas de máquinas físicas e virtuais. Por fim, no capítulo 6, são feitas as considerações finais e comentários sobre o trabalho. Neste capítulo, também são apresentadas propostas para trabalhos futuros no âmbito deste trabalho.

2 FUNDAMENTOS: VIRTUALIZAÇÃO E MONITORAMENTO

Neste capítulo, serão apresentados conceitos e estratégias fundamentais para o correto entendimento deste trabalho. Inicialmente, serão apresentados conceitos sobre virtualização e o ambiente de virtualização Citrix XenServer, que é empregado no ambiente monitorado. Logo após, serão apresentados conceitos sobre monitoramento de ambientes computacionais. Serão discutidas as estratégias adotadas no monitoramento de máquinas virtuais. A estrutura e os componentes da ferramenta de monitoramento Nagios, que é a ferramenta empregada no ambiente que será monitorado, serão apresentados, sendo que o foco será dado àqueles componentes que contribuem para a solução do problema.

2.1 Virtualização

Virtualização é um conceito antigo na computação e data dos anos 60, onde era utilizado para denominar sistemas computacionais que forneciam uma abstração em relação ao hardware. Nos anos 90, este conceito apareceu novamente com a introdução da linguagem de programação Java (CARISSIMI, 2009). Nota-se que estas duas tecnologias são bem distintas e utilizam o mesmo conceito de virtualização: abstração do hardware ou plataforma. Virtualização, portanto, é uma abstração que fornece uma visão de um sistema computacional completo. No caso de Java, o sistema operacional também é abstraído atribuindo à Java a característica de ser independente de plataforma. As grandes vantagens no uso de virtualização são conseqüência direta desta independência de plataforma. Dentre estas vantagens estão tolerância a falhas, disponibilidade, segurança, manutenibilidade, as quais são possíveis pelo fato de uma máquina virtual poder ser executada sobre qualquer hardware suportado pela plataforma de virtualização (SMITH; NAIR, 2005). Se este ambiente é formado por várias máquinas físicas basta que as máquinas virtuais afetadas por uma falha no hardware sejam migradas para outras máquinas físicas. Estas migrações podem ser feitas também quando uma determinada máquina virtual exige muito processamento da máquina física, prejudicando as demais máquinas hospedadas por esta máquina física. As máquinas afetadas podem ser migradas para outra máquina física ociosa, provendo ao ambiente a característica chamada de balanceamento de carga.

Muitos ambientes computacionais apresentam a filosofia de “um serviço por máquina”. Desta forma, cada serviço possui um hardware dedicado, o qual fornece isolamento e independência de configuração. Por outro lado, essa filosofia acaba aumentando drasticamente o número de máquinas em um sistema computacional. Além disso, esses serviços não utilizam a capacidade total da máquina onde estão instalados, havendo desperdício de investimento, recursos, energia, etc. A virtualização está sendo

utilizada para resolver este problema. Com plataformas de virtualização, pode-se estender esta filosofia para “um serviço por máquina virtual” e diversas máquinas virtuais em uma mesma máquina física. Cada máquina virtual executará seu próprio sistema operacional, com as configurações adequadas ao serviço que hospeda, atingindo os mesmos objetivos da filosofia anterior: isolamento e independência de configuração

2.2 Citrix XenServer

XenServer é oferecido pela Citrix e é baseado no Hypervisor de código aberto Xen. A versão gratuita do XenServer oferece migrações de máquinas virtuais sem que seja necessário desligá-las, mas balanceamento de carga e alta disponibilidade são oferecidas apenas nas versões pagas. Para que estas migrações funcionem, o Citrix XenServer necessita de um armazenamento centralizado, o qual é compartilhado entre as máquinas físicas do cluster. Qualquer máquina física que visualize este armazenamento centralizado pode hospedar uma máquina virtual que esteja neste armazenamento compartilhado. Também podem ser utilizadas máquinas virtuais armazenadas localmente, mas sem migrações, balanceamento de carga e alta disponibilidade.

Citrix XenServer utiliza uma estratégia de mestre/escravo para orquestrar o cluster. Toda a informação sobre o cluster e o relacionamento entre máquinas virtuais e máquinas físicas pode ser acessado através de qualquer máquina física.

2.3 Monitoramento

Monitoramento é o processo de obter informações sobre elementos de um sistema computacional. Estas informações ajudam a entender a situação do sistema, sua configuração, estatísticas de uso e desempenho, informações sobre erros e sobre a topologia do sistema. O processo de monitoramento depende de técnicas para coletar, processar, armazenar e disponibilizar estas informações. Entretanto, a variedade de elementos que compõem um sistema computacional exigirá técnicas adequadas para cada classe de elemento (VERMA, 2009).

Em um sistema computacional, um dos elementos mais importantes são os servidores. Para o monitoramento de servidores, alguns aspectos são importantes, tais como a utilização do hardware como CPU (Unidade Central de Processamento), memória, disco ou rede, que fornecem métricas diretas para verificar seu desempenho. Entretanto, servidores são máquinas que fornecem algum tipo de serviço, logo, é natural que a situação destes serviços seja monitorada. Cada serviço deve ser monitorado de maneira que suas peculiaridades sejam consideradas. Por exemplo, para um servidor web, é interessante que seja monitorado o tempo de resposta para a requisição de uma página ou o número de páginas fornecidas por segundo. Já para um servidor de e-mail, seria interessante que fosse monitorado o número de mensagens processadas por segundo ou a quantidade de mensagens enfileiradas para entrega. Diversas ferramentas estão disponíveis para o monitoramento de servidores e serviços, tais como Monit, Cacti, Zabbix e Nagios. Muitas destas ferramentas trazem rotinas prontas para o monitoramento de serviços amplamente utilizados, restando para o administrador apenas compor sua ferramenta de monitoramento como blocos de construção.

2.4 Monitoramento de Máquinas Virtuais

Diferentes abordagens são utilizadas para o monitoramento de máquinas virtuais. Algumas destas abordagens monitoram o Virtual Machine Monitor (VMM) (SHAO; JIN; LU, 2009; PAYNE; CARBONE; LEE, 2007) e outras, baseadas em introspecção, obtêm as informações que precisam através da análise de memória da máquina virtual. Estas duas abordagens são dependentes de plataforma e isto torna mais difícil a implementação e integração com ferramentas de monitoramento legadas. A abordagem mais comum é monitorar a máquina virtual como uma máquina física. A implementação desta abordagem é simples porque o administrador do sistema apenas incorpora as máquinas virtuais dentro da ferramenta de monitoramento legada. Por outro lado, esta abordagem ignora completamente as informações sensíveis sobre o ambiente virtualizado, como a saúde da máquina física e a informação de relacionamento entre máquinas virtuais e físicas.

O monitoramento baseado em introspecção é a estratégia mais sofisticada. Nesta abordagem, as variáveis são coletadas de fora da máquina virtual por um sistema instalado no sistema operacional da máquina física. Estes sistemas inspecionam o espaço de memória da máquina virtual e descobrem o valor destas variáveis. Esta abordagem está sendo utilizada para detecção de intrusão. Alguns sistemas propostos apenas coletam estas variáveis e enviam uma notificação para o administrador do sistema (FRASER; EVENSON; ARBAUGH, 2008). Outras, detectam a invasão e corrigem as vulnerabilidades introduzidas pelo invasor (BAIARDI; SGANDURRA, 2007).

Os ambientes virtualizados possuem suas próprias ferramentas de monitoramento que são dependentes de plataforma como o XenCenter para o Citrix XenServer e o VMware vCenter para o VMware vSphere. Estas soluções podem ser utilizadas em conjunto com uma ferramenta de monitoramento integradora como o DataGraph (HUANG et al, 2009). O DataGraph consolida as informações de monitoramento de outras ferramentas de monitoramento dentro de sua base de dados e permite uma visualização do ambiente inteiro em uma interface de monitoramento única. Para o monitoramento de máquinas virtuais, DataGraph pode ser utilizado para integrar informações coletadas das ferramentas de monitoramento específicas de um ambiente virtualizado com a informação de monitoramento de ferramentas de monitoramento legadas.

Ferramentas de monitoramento adaptativo podem ser utilizadas em ambientes de corporações onde reconfigurações ocorrem com freqüência (MACHIDA; KAWATO; MAENO, 2007). Estas ferramentas de monitoramento são projetadas especificamente para trabalhar com ambientes virtualizados e já são fornecidas com funções embutidas para tratar migrações e medidas de carga nestes ambientes. Mas esta solução deve ser integrada com outra ferramenta de monitoramento, assim como as ferramentas fornecidas pelas plataformas de virtualização, para satisfazer todos os outros objetos monitorados e isto pode tornar seu uso proibitivo.

2.5 Ferramenta de monitoramento Nagios

Nagios é uma ferramenta de monitoramento de código aberto amplamente utilizada em corporações. Sua configuração é baseada em objetos hierarquicamente organizados. No topo, estão os “hostgroups”. Que contém os “hosts”. Que contém os “services”. Os “hostgroups” trabalham em conjunto com os “hosts templates” para minimizar o

retrabalho em reconfigurações da ferramenta de monitoramento. Por exemplo, o período que um host deve ser monitorado, imagem destes hosts no mapa, o intervalo de tempo para notificações é configurado nos “hosts templates” e os “hosts” herdam estes valores. Também são fornecidos “service templates” para simplificar a configuração dos “services”. Estes “services templates” armazenam quais tipos de checagem um serviço aceita, qual o intervalo entre as checagens, grupos de contatos, e o período que os serviços devem ser monitorados.

Nagios utiliza dois tipos de checagem: ativo e passivo. Quando a iniciativa da checagem é tomada pelo processo do Nagios esta checagem é chamada ativa. Outros processos podem informar para o Nagios o estado de um serviço, quando isto acontece, a checagem é chamada passiva.

Para fazer uma checagem ativa, o Nagios executa um comando configurado na definição do serviço monitorado, captura a saída do comando e a armazena em suas estruturas de dados. Estes comandos utilizam plug-ins que podem ser desenvolvidos por qualquer um e incorporados na configuração do Nagios. Estes comandos são executados localmente no servidor Nagios. Para executar estes comandos e capturar a informação de estado de máquinas remotas, os administradores de sistema podem utilizar o Nagios Remote Plug-in Executor (NRPE). O NRPE é um plug-in instalado no servidor Nagios e um daemon instalado na máquina monitorada.

Um processo que deseja informar o estado de um serviço para o Nagios através de uma checagem passiva deve escrever seus resultados no “External Command File”. O Nagios processa os dados deste arquivo em um intervalo de tempo configurado. Isto é possível apenas para processos locais. O daemon Nagios Service Check Acceptor (NSCA) permite que hosts remotos coloquem seus resultados para as checagens de serviço no “External Command File”.

Nagios oferece um intermediador de eventos chamado Nagios Event Broker (NEB) que permite que qualquer um escreva um módulo para trabalhar com este intermediador. O NEB cria canais dedicados para cada tipo de evento e os módulos requisitam o registro nos canais que lhe interessam. Quando determinado evento ocorre, o Nagios procura os módulos que registraram interesse no canal do evento, chama o módulo e passa uma estrutura de dados com informações sobre o evento. O controle do processo do Nagios é passado para o módulo e ele tem acesso a todas as estruturas de dados do Nagios. Como Nagios é um simples laço infinito, os módulos devem processar rapidamente e retornar o controle para o processo Nagios para não afetar o desempenho de todo o sistema Nagios.

3 SOLUÇÃO PROPOSTA

Atualmente, os administradores de rede possuem sobre sua responsabilidade uma quantidade muito grande de máquinas e serviços. Neste cenário, o administrador necessita de ferramentas que auxiliem e automatizem suas tarefas. Estas ferramentas devem disponibilizar rapidamente uma visão coerente da situação dos serviços sob a responsabilidade do administrador.

Outra realidade é a utilização de virtualização como solução simples para obter alta disponibilidade e utilização eficiente de recursos. Com a utilização de virtualização, uma determinada organização pode utilizar diversas máquinas virtuais sobre apenas uma máquina física diminuindo o desperdício de recursos que ficariam ociosos. Recentemente, algumas plataformas de virtualização têm oferecido suporte automatizado para alta disponibilidade. Estas plataformas detectam falhas e sobrecargas e reconfiguram o sistema migrando máquinas virtuais entre as máquinas físicas disponíveis distribuindo a carga total.

A abordagem mais simples, que consiste em monitorar máquinas virtuais olhando para elas como máquinas físicas, ignora o fato de que a máquina virtual é executada por uma máquina física. Para melhorar esta solução, seria interessante que a saúde da máquina física seja considerada no monitoramento da máquina virtual. Então, seria interessante, para os administradores, que em sua ferramenta de monitoramento fossem visualizadas informações da máquina física juntamente com as informações da máquina virtual. Afinal, as informações da máquina física (como uso de CPU, memória, disco, etc) influenciam diretamente no desempenho da máquina virtual. Entre as informações da máquina virtual, está a situação de seus serviços e o monitoramento dos recursos virtuais que lhe foram atribuídos.

Deve-se notar que, mesmo quando o monitoramento mostra que o uso do recurso virtual está dentro do esperado, não se pode afirmar que esta máquina virtual não está enfrentando problema de escassez de recursos. A concorrência pelos recursos físicos pode estar impedindo que a máquina virtual adquira recursos virtuais. Por exemplo, o monitoramento da CPU da máquina virtual pode indicar que ela está usando 5% de sua carga, entretanto, a máquina física está utilizando 100 % de sua CPU e consegue atribuir a esta máquina virtual apenas 5 % do que ela teria direito.

De certa forma, o administrador poderia configurar sua ferramenta de monitoramento para adicionar as informações de uso dos recursos físicos, informando em que máquina física esta ferramenta deveria colher estas variáveis. Ele escreve um comando de checagem e informa, através de um argumento, onde a máquina virtual está sendo executada e obtém a informação da saúde da máquina física. Mas toda vez que uma máquina virtual é migrada, os argumentos para o comando de checagem devem ser

corrigidos. Nesta solução, o administrador de sistema deve monitorar as migrações e corrigir seus comandos de checagem.

Outro administrador de sistema pode utilizar ferramentas de monitoramento distintas para cada tipo de objeto monitorado. Por exemplo, monitora os serviços de uma máquina virtual com uma ferramenta de monitoramento, como o Nagios, e monitora o relacionamento entre máquinas virtuais com suas máquinas físicas e sua saúde com uma ferramenta fornecida pelo ambiente de virtualização como o XenCenter.

A adoção da primeira solução ficaria prejudicada com a utilização das atuais plataformas que migram as máquinas virtuais entre servidores físicos sem intervenção do administrador. Ou seja, o administrador nem sempre sabe exatamente onde determinada máquina virtual está hospedada e, conseqüentemente, não conseguiria configurar manualmente a ferramenta de monitoramento. Além disso, a configuração da ferramenta de monitoramento rapidamente ficaria inconsistente a cada migração que ocorresse no ambiente monitorado. Já a segunda solução torna mais complexa a rotina diária dos administradores que não visualizam seu ambiente por inteiro e devem conhecer quais informações estão em cada ferramenta de monitoramento.

Neste cenário, a ferramenta de monitoramento deve ser sensível à dinâmica natural destes ambientes sem a intervenção dos administradores. A ferramenta deve fornecer informações sobre a topologia do ambiente e apresentar informações sobre os recursos da máquina física, juntamente com as informações sobre os recursos e serviços da máquina virtual. Desta maneira, o administrador pode consolidar as informações pertinentes a um ambiente de virtualização em apenas uma ferramenta.

Além dessas características funcionais, outras características qualitativas devem ser consideradas no desenvolvimento de uma solução, tais como manter a ferramenta de monitoramento disponível a maior parte do tempo possível, não ser prejudicado por falhas que ocorram no ambiente monitorado, deve-se utilizar ao máximo de componentes disponíveis na ferramenta de monitoramento e deve-se utilizar a mesma interface de visualização que a ferramenta de monitoramento utiliza.

Muitas informações nas ferramentas de monitoramento somente são lidas, a partir de seus arquivos de configuração, em sua inicialização. Logo, é possível de se imaginar que uma solução proposta escreva novas informações nos arquivos de configuração e tenha que reinicializar a ferramenta de monitoramento para que estas informações sejam disponibilizadas na interface. Se estas reinicializações não forem minimizadas, a ferramenta de monitoramento pode ter sérios problemas de disponibilidade.

A solução proposta não pode ser sensível a falhas no ambiente monitorado. Uma solução proposta poderia utilizar um servidor do cluster como sendo responsável por informar as informações do cluster para o Nagios, já que todas as máquinas do cluster Citrix XenServer conhecem todo o cluster. No caso de uma falha neste servidor, toda a lógica do monitoramento dos servidores virtuais seria comprometida. Mesmo utilizando várias destas máquinas físicas, o monitoramento pode ser afetado, como por exemplo, se todas as máquinas que utiliza forem afetadas por algum problema de conectividade.

A solução proposta deverá utilizar ao máximo os componentes já disponibilizados pela ferramenta de monitoramento. Ou seja, a instalação da solução proposta deve ser minimalista. Os administradores têm sérias restrições quanto à instalação de novos pacotes e programas nos servidores. Cada pacote ou programa pode inserir novas vulnerabilidades ou falhas de software no ambiente monitorado. A ferramenta de

monitoramento não pode inserir mais possibilidades de falhas e vulnerabilidades no ambiente. E por outro lado, uma solução em que diversos componentes adicionais devem ser instalados pode cair em desuso pela complexidade de sua instalação.

A solução proposta deve utilizar a mesma interface que a ferramenta de monitoramento. Em primeiro lugar, pela restrição anterior, para que uma nova interface não tenha que ser instalada. Além disso, os administradores já estão habituados na utilização desta interface e podem ter tarefas automatizadas baseadas nesta interface. Outra interpretação desta restrição é que deve existir apenas uma interface de visualização na ferramenta de monitoramento. Ou seja, o administrador não deve utilizar uma interface para um determinado conjunto de informações e outra interface para outro conjunto de informações. A solução proposta deve integrar estes conjuntos de informações na mesma interface.

A solução que será proposta integra seus objetivos dentro de uma ferramenta de monitoramento única. Esta solução proposta integra a abordagem mais simples de monitorar máquinas virtuais como máquinas físicas com a abordagem de coletar informações do VMM. O foco da solução é fazer melhoramentos na ferramenta de monitoramento Nagios, a fim de monitorar máquinas virtuais do ambiente virtualizado Citrix XenServer.

Os objetivos da solução são integrar informação de monitoramento da máquina física com as informações de monitoramento da máquina virtual e colocar na ferramenta de monitoramento a informação do relacionamento entre máquinas virtuais e físicas. Estes objetivos devem ser alcançados considerando migrações de máquinas virtuais entre máquinas físicas.

Para alcançar este objetivo, pode-se dividir o problema em algumas etapas, tais como obter a lista de máquinas virtuais de cada máquina física, atualizar a interface do Nagios para mostrar estas informações e agrupar as informações sobre o estado da máquina física, juntamente com as informações da máquina virtual.

Para a primeira etapa, poderia ser utilizada uma Cron de sistema, no servidor Nagios, para executar um comando em cada máquina física. Poderia utilizar o Secure Shell (SSH) para disparar este comando na máquina remota. O primeiro ponto negativo desta solução é descobrir quais máquinas físicas contatar. Este comando poderia utilizar uma lista fixa ou, então, outro comando poderia recuperar esta lista de uma das máquinas físicas, já que cada máquina física do cluster XenServer da Citrix conhece o estado de todo o cluster. Uma lista fixa precisa ser atualizada pelo administrador toda vez que se adiciona ou retira uma máquina física do cluster. Neste caso, o administrador deverá lembrar-se que deve atualizar esta lista. A segunda estratégia utiliza um servidor fixo para contatar, o que pode deixar o monitoramento inconsistente em caso de falha nesta máquina física. Esta estratégia poderia ser melhorada colocando uma lista de máquinas a serem contatadas, mas recorre-se aos problemas da primeira solução.

Ainda na primeira etapa, é interessante que esta informação, a lista de máquinas virtuais de uma máquina física, fique visível para o administrador. Desta forma, faz muito sentido colocá-la como um novo serviço na configuração do Nagios que se refere à máquina física. Esta informação fica disponível nas estruturas de dados do Nagios e pode ser utilizada nas etapas posteriores.

Um serviço necessita de um comando de checagem para a estratégia, utilizando checagem ativa, que é executado a cada intervalo de checagem. No caso, como se trata

de uma checagem remota, no servidor Nagios utiliza-se o plug-in “check_nrpe” para contatar o daemon NRPE na máquina remota que chama o plug-in “check_xen_virtual_machines”, instalado na máquina física, o qual recupera a lista de máquinas virtuais daquela máquina física.

A interface do Nagios não utiliza diretamente as estruturas de dados internas para atualizar a sua interface. Ele atualiza um arquivo chamado “object_cache_file” que é processado pela interface. Para a segunda etapa, uma possível solução seria capturar a lista armazenada no novo serviço, adicionado à configuração da máquina física no Nagios, e atualizar este arquivo, o “object_cache_file”, com as informações do serviço. O primeiro problema desta solução é que novamente deve-se ter uma lista de máquinas físicas para poder consultar o conteúdo do serviço, visto que a macro $SERVICEOUTPUT$ precisa do nome da máquina e do nome do serviço como argumento. Outro problema é definir em que momento este comando será chamado. Uma estratégia do tipo “polling” é muito onerosa por utilizar constantemente a CPU em um laço infinito. Uma estratégia mais econômica seria utilizar um Cron de sistema que, periodicamente, executa este comando e atualiza o conteúdo do arquivo “object_cache_file”. Entretanto, esta solução insere mais um tempo de latência para a solução o que aumenta o tempo em que a interface levará para mostrar a situação real do cluster. Uma boa solução teria que atualizar o arquivo tão logo uma nova lista de máquinas virtuais de uma máquina física seja conhecida. Isto é possível através de um módulo para o NEB, que recebe notificações quando determinado tipo de evento ocorre no Nagios. O módulo “detect_host_change” escuta os eventos do canal “SERVICE_STATE_CHANGE” e alcança este objetivo.

Para alcançar o terceiro objetivo, será necessário adicionar serviços à configuração da máquina virtual para incorporar o estado dos serviços de sua máquina física. O comando de checagem destes serviços deverá utilizar um plug-in que descubra qual máquina física deve contatar e obter o estado do serviço desta máquina. Novamente, este plug-in deveria consultar a lista de máquinas virtuais de cada máquina física. Deveria possuir uma lista de máquinas físicas para verificar a lista, armazenada no serviço, de cada máquina física e determinar onde esta máquina virtual está executando. O módulo “detect_host_change”, para o NEB, resolve este problema gravando um arquivo com o nome desta máquina virtual contendo o nome da máquina física que a executa. Assim, o plug-in “check_physical” apenas consulta este arquivo e contata a máquina física correta.

Como a maioria dos problemas para as soluções propostas é em relação às listas de máquinas virtuais, uma solução sem listas foi elaborada. Nesta solução, cada máquina virtual teria um serviço adicionado à sua configuração do Nagios. Este serviço, que poderia denominar-se “physical_host”, armazenaria o nome da máquina física que hospeda esta máquina virtual. O módulo NEB poderia capturar o conteúdo deste serviço e modificar o arquivo para a interface e o plug-in “check_physical” poderia ler o conteúdo deste serviço, ao invés de ler os arquivos gerados pelo módulo NEB. Entretanto, esta solução impossibilita o uso de uma estratégia utilizando checagem ativa, pois para descobrir a máquina física que hospeda esta máquina virtual, deveria consultar alguma máquina física do cluster. Para que isso fosse possível, este plug-in deveria possuir uma lista fixa de máquinas do cluster para contatar. O que se recorre aos problemas anteriores relacionados às listas fixas. Ou seja, a informação deste serviço deveria ser informada passivamente ao Nagios, onde a iniciativa de enviar o conteúdo deste serviço partiria de uma máquina física do cluster. Esta solução também aumenta o

número de serviços adicionados ao Nagios, visto que cada máquina virtual deverá possuir um serviço para armazenar o nome da máquina física. Este aspecto prejudica o desempenho do Nagios e aumenta o arquivo "object_cache_file", o qual deve ser processado pelo módulo NEB. Além disso, aumenta o esforço de configuração necessário pelo administrador comparando-se com a solução empregada, onde basta o administrador adicionar um serviço para cada máquina física.

3.1 Plug-in "check_xen_virtual_machines" para o Nagios

O primeiro objetivo é obter a lista de máquinas virtuais que uma máquina física do cluster Citrix XenServer está hospedando. Na máquina física, é instalado um plug-in chamado "check_xen_virtual_machines" para ser utilizado com o NRPE. Este plug-in obtém esta informação do hypervisor do Citrix XenServer. No servidor Nagios, um novo serviço é adicionado à configuração da máquina física no Nagios. Este serviço, chamado "virtual_machines", vai armazenar a lista de máquinas virtuais executadas atualmente pela máquina física.

Este plug-in foi desenvolvido para trabalhar na estratégia utilizando tanto a checagem ativa como a passiva. Quando ele é executado sem parâmetros, assume que está atuando em uma estratégia com checagem ativa. Quando chamado com o parâmetro "--nsca", assume que está atuando em uma estratégia passiva. E quando utilizados os parâmetros "--nsca --aggressive", assume que está atuando em uma estratégia utilizando checagem passiva agressiva.

No caso da utilização de alguma das estratégias de checagem passiva, este plug-in armazena o comando utilizado para enviar o resultado da checagem em um arquivo em "/tmp". Este comando é a chamada enviada ao daemon NSCA para informar passivamente o resultado de uma checagem de serviço ao Nagios. Na próxima execução, ele compara o novo comando que será enviado ao Nagios com aquele armazenado em "/tmp". Se o comando for idêntico, ele não envia este comando para o Nagios. Se o arquivo em "/tmp", que armazena o comando previamente enviado, foi criado há mais de 10 minutos, o plug-in força um comando de checagem ao Nagios, mesmo que este comando seja igual ao comando informado previamente. Isto para manter as informações atualizadas no Nagios caso ele tenha sido reinicializado.

No caso da utilização de estratégia por checagem ativa, o plug-in coloca a lista de máquinas virtuais daquela máquina física na saída padrão, adicionando o prefixo "OK", que é uma saída padronizada para plug-ins do Nagios. Se não houver máquinas virtuais sendo executadas por esta máquina física, o plug-in retorna à saída padrão "Warning – none" para indicar que pode haver algum problema na máquina física que a impede de hospedar máquinas virtuais. O administrador pode saber o motivo pelo qual aquela máquina física não possui máquinas virtuais e, neste caso, pode ignorar o alerta. Apesar de a saída indicar textualmente o estado do serviço, "OK" ou "Warning", o Nagios espera um código de retorno do plug-in para indicar o estado do serviço. A informação que realmente indica a situação do serviço é o código retornado pela execução do plug-in.

A Figura 3.1 mostra como fica a interface do Nagios com o serviço adicional, chamado "virtual_machines" e seu conteúdo. Deve-se notar que o conteúdo deste serviço é idêntico tanto para checagem ativa como passiva.

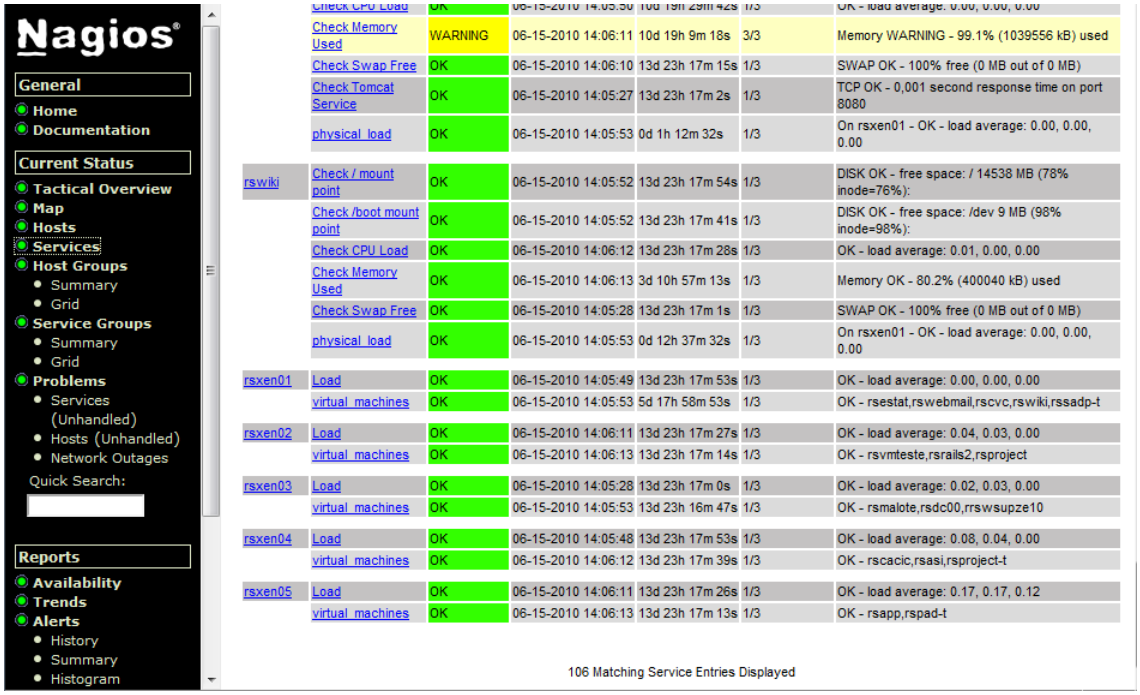

Figura 3.1: Interface do Nagios mostrando o serviço “virtual_machines”.

3.2 Módulo “detect_host_change” para o NEB

Este módulo é responsável por capturar a lista de máquinas virtuais de uma máquina física, atualizar as informações nas estruturas de dados internas do Nagios, atualizar a interface do Nagios com as novas informações e informar ao plug-in “check_physical” onde cada máquina virtual está sendo executada.

A primeira tarefa é obtida escutando o canal “SERVICE_STATE_CHANGE” do NEB. Neste canal, ele captura a lista de máquinas virtuais e, a partir desta lista, consegue atingir os demais objetivos. Em seguida, ele altera as informações de parentesco entre as máquinas virtuais e físicas nas estruturas internas do Nagios. Se este módulo detectar que houve uma mudança no relacionamento entre estas máquinas, ele atualiza a interface do Nagios com este novo relacionamento e grava um arquivo para cada máquina virtual. Cada arquivo recebe o nome da máquina virtual e, em seu conteúdo, está o nome da máquina física que a executa.

Na Figura 3.2, a interface do Nagios mostra o relacionamento entre as máquinas físicas e virtuais. Neste caso, o administrador não necessitou informar estes relacionamentos que foram atribuídos pelo módulo NEB. Caso o módulo fosse desativado, a interface mostraria todas as máquinas lado a lado.

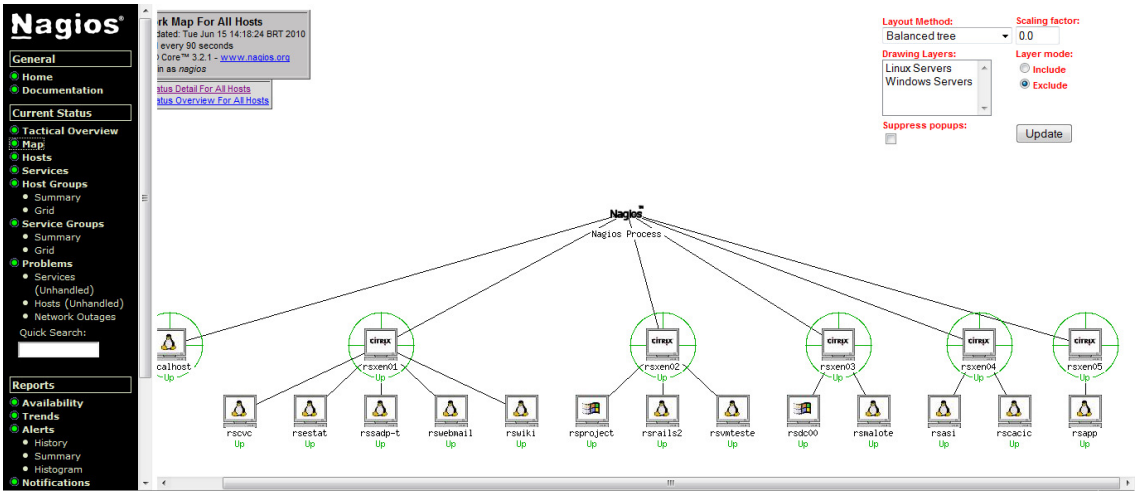

Figura 3.2: Mapa de relacionamento entre as máquinas virtuais e físicas.

3.3 Plug-in “check_physical” para o Nagios

Este plug-in é utilizado para obter o estado dos serviços da máquina física. O módulo NEB cria um arquivo com o nome da máquina virtual e, no conteúdo deste arquivo, encontra-se o nome da máquina física que a hospeda. Este plug-in lê a informação neste arquivo e chama o plug-in “check_nrpe”, que é responsável por contatar o daemon NRPE da máquina física. Captura a saída gerada por este plug-in e o código de retorno, coloca a mesma saída em sua saída padrão e termina sua execução com o código de retorno recebido.

Na Figura 3.3, a interface do Nagios mostra o serviço “physical_load”, o qual utiliza o plug-in “check_physical” para obter a carga atual na máquina física. Na saída padrão dos serviços que utilizam este plug-in, é adicionado o nome da máquina onde o plug-in coletou esta informação. Deve-se notar que “physical_load” é apenas um exemplo. Se outras métricas são capturadas pelo Nagios para a máquina física, como CPU, memória, uso de disco ou rede, estas métricas também podem ser incorporadas às informações das máquinas virtuais da mesma maneira como foi feito para “physical_load”.

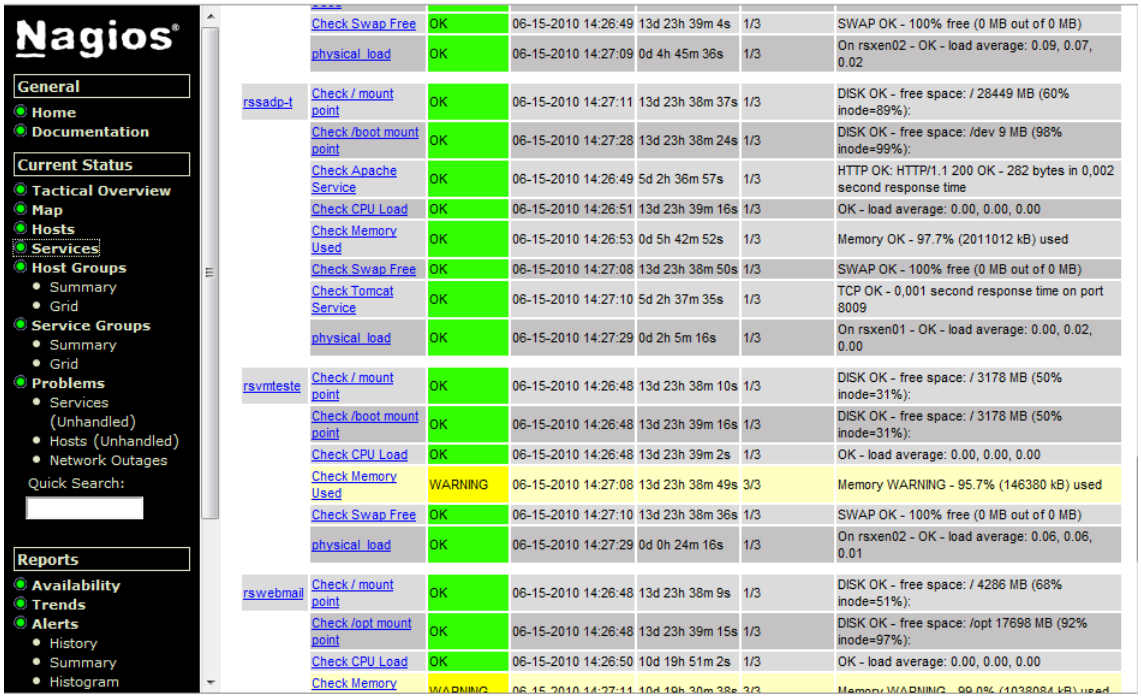

Figura 3.3: Interface do Nagios mostrando o serviço “physical_load”.

3.4 Estratégia de Checagem Ativa

Todos os componentes apresentados anteriormente serão utilizados no caso da utilização de checagem ativa. Nesta estratégia, um comando de checagem é colocado nas configurações do Nagios que definem o serviço. No caso, um serviço chamado “virtual_machines” deverá ser adicionado à configuração da máquina física no servidor Nagios. O comando de checagem deste serviço deve utilizar o plug-in “check_nrpe”, no servidor Nagios, para contatar o daemon NRPE na máquina física que executa o plug-in “check_xen_virtual_machines” e obtém a lista de máquinas virtuais executando na máquina física. Este funcionamento é ilustrado pela Figura 3.4.

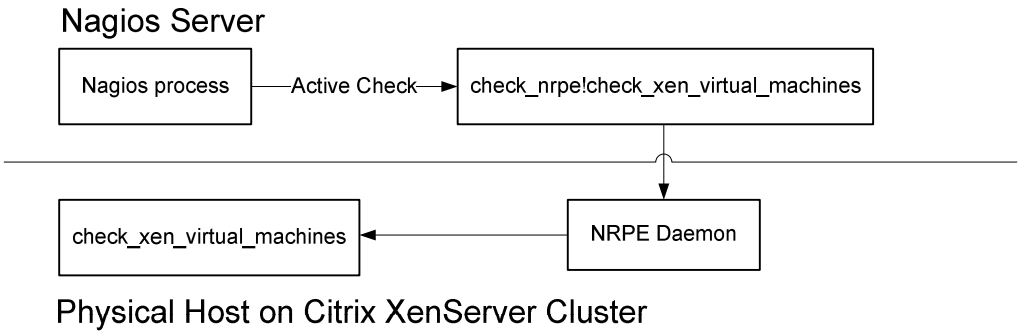

Figura 3.4: Plug-in “check_xen_virtual_machines” em checagem ativa.

O módulo NEB, chamado “detect_host_change”, intercepta a lista de máquinas virtuais cada vez que o Nagios recebe o conteúdo do serviço “virtual_machines”. Ao capturar esta lista, atualiza as estruturas de dados do Nagios, as suas próprias estruturas, os arquivos utilizados pelo plug-in “check_physical” e atualiza o arquivo “object_cache_file” para mostrar as mudanças impostas por esta nova lista. Isto é ilustrado pela Figura 3.5.

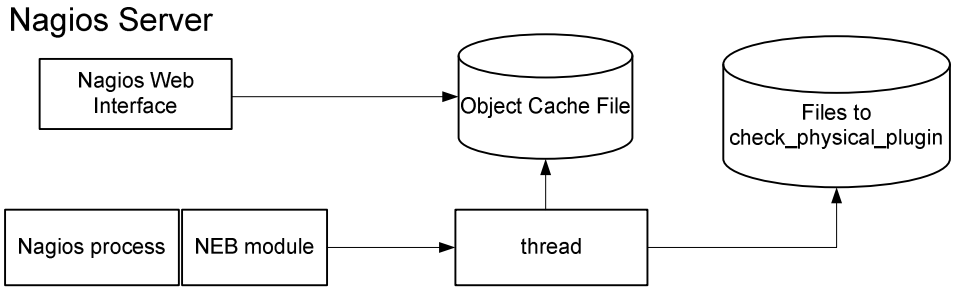

Figura 3.5: Módulo NEB.

O plug-in “check_physical” tem um trabalho simples a ser feito agora. Ele simplesmente lê o arquivo armazenado pelo módulo NEB e descobre onde aquela máquina virtual está executando. Então, ele chama o plug-in “check_nrpe” para obter a informação de estado da máquina física. Por exemplo, pode ser adicionado um novo serviço à configuração da máquina virtual no Nagios, o qual poderia chamar-se “physical_load”, que utilizaria o plug-in “check_physical” e o plug-in “check_load” na máquina física. A Figura 3.6 ilustra isso.

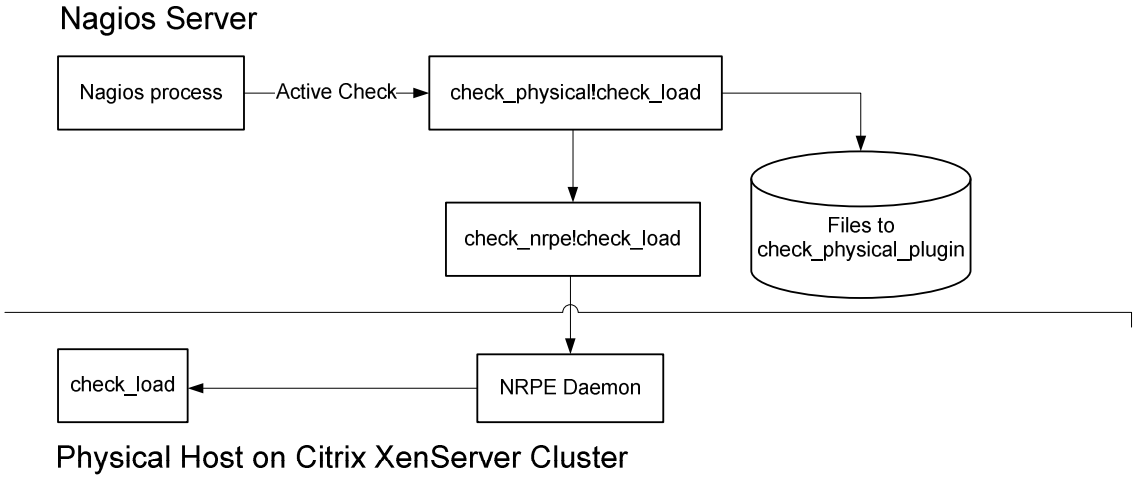

Figura 3.6: Plug-in “check_physical”.

3.5 Estratégia de Checagem Passiva

A estratégia de checagem passiva consiste em a máquina física informar ao servidor Nagios quais máquinas virtuais estão sendo executadas por ela. O mesmo plug-in “check_xen_virtual_machines”, desenvolvido para a estratégia de checagem ativa, pode ser utilizado. Na estratégia de checagem ativa, este plug-in é chamado pelo daemon NRPE que captura sua saída e a retorna para o processo Nagios. Na estratégia de checagem passiva, o Cron do sistema deve chamar este plug-in. Agora, o plug-in comporta-se como um script. O plug-in utiliza o comando “send_nsca” para contatar o daemon NSCA no servidor Nagios. O funcionamento do plug-in “check_xen_virtual_machines”, nesta estratégia, é ilustrado pela Figura 3.7.

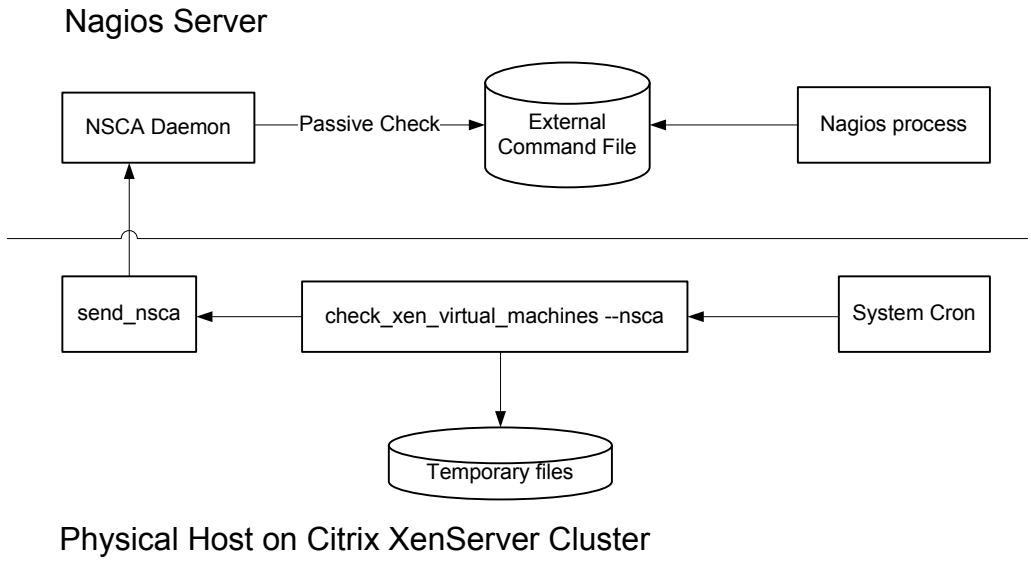

Figura 3.7: Plug-in “check_xen_virtual_machines” em checagem passiva.

Da mesma forma que na checagem ativa, o módulo NEB interceptará o conteúdo do serviço e executará as mesmas tarefas. O mesmo vale para o plug-in “check_physical” que utiliza os arquivos gravados pelo módulo NEB.

3.6 Estratégia de Checagem Passiva Agressiva

Cada máquina física, no cluster Citrix XenServer, conhece o estado de todo o cluster. Uma estratégia intermediária pode ser utilizada que consiste em utilizar o plug-in “check_xen_virtual_machines” para informar para o Nagios o estado de todo o cluster de uma só vez. Isto é feito utilizando o plug-in no modo “--nsca--aggressive”. A primeira máquina física no cluster que detectar uma mudança no relacionamento entre máquinas virtuais e físicas informa esta mudança ao Nagios. Esta estratégia intermediária aumenta a utilização de rede para tentar minimizar o tempo de detecção de uma mudança. Assim como na outra estratégia utilizando checagem passiva, os demais componentes da solução são utilizados da mesma forma. O funcionamento do plug-in “check_xen_virtual_machines”, nesta estratégia, é ilustrado pela Figura 3.8.

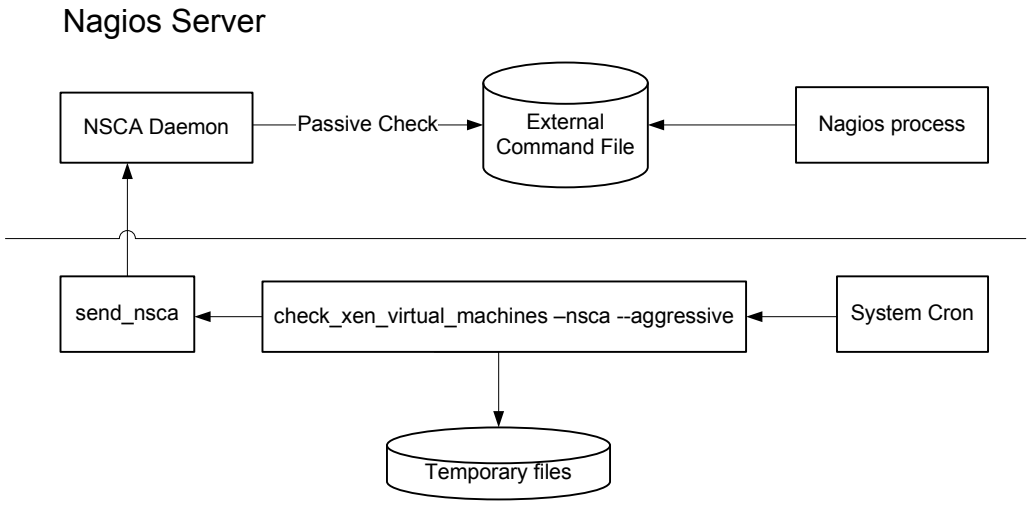

Figura 3.8: Plug-in “check_xen_virtual_machines” em checagem passiva agressiva.

4 IMPLEMENTAÇÃO

Este capítulo visa fornecer detalhes específicos da implementação dos componentes da solução proposta. Estes componentes foram utilizados na etapa de avaliação e foram empregados no ambientes real que foi monitorado. Por este motivo, serão apresentados alguns detalhes de instalação e configuração destes componentes.

4.1 Plug-in Check_xen_virtual_machines para o Nagios

Este plug-in foi desenvolvido em Perl. Esta linguagem foi escolhida porque grande parte dos plug-ins para o Nagios é desenvolvida em Perl. Ou seja, provavelmente o interpretador Perl estará disponível na máquina física, não requerendo, então, a instalação de pacotes adicionais.

Este plug-in é específico para o Citrix XenServer, pois captura informações diretamente do hypervisor através de comandos disponibilizados pela plataforma. Ele executa o comando "xe host-list" para capturar o id e o nome de todas as máquinas físicas do cluster. Ele armazena estas informações em uma estrutura hash onde a chave é o nome da máquina física e a informação armazenada pelo hash é o uuid desta máquina física.

Depois, executa o comando "xe vm-list" para capturar todas as máquinas virtuais do cluster que estão com a variável "power-state" ajustada para "running". Desta forma, máquinas virtuais pausadas, desligadas, ou em processo de migração, não são informadas ao Nagios enquanto não estiverem com seu estado "running". O Citrix XenServer não coloca o nome da máquina virtual em suas descrições. Entretanto, existe uma variável que pode ser preenchida pelo administrador chamada "name-label". O plug-in considera a primeira palavra da variável "name-label" como sendo o nome da máquina virtual. Ele captura o uuid da máquina física que executa esta máquina virtual no campo "resident-on" da máquina virtual. Ele armazena estas informações em outra estrutura hash onde a chave é o uuid da máquina física e a informação armazenada é a lista de máquinas virtuais daquela máquina física. Ou seja, a cada máquina virtual retornada pelo comando "xe vm-list" o plug-in consulta a estrutura hash para descobrir se o uuid da máquina física que executa esta máquina virtual já está cadastrado na estrutura hash. Se já pertence ao hash, outra máquina virtual desta máquina física já foi cadastrada, então, o nome da nova máquina virtual é concatenado ao nome das máquinas que já estão cadastradas. Caso não pertença, esta entrada hash é definida apenas com o nome desta máquina virtual.

Se a estratégia por checagem ativa estiver sendo utilizada, o plug-in procura no hash, onde o nome da máquina física é a chave, o uuid da máquina física que está executando o plug-in. Ele acessa o hash utilizando o hostname desta máquina. Logo

depois, utiliza este uuid para consultar o hash que contém o nome das máquinas virtuais que a máquina com este uuid está executando. Se o hash possui este uuid cadastrado, então a máquina física hospeda máquinas virtuais. O plug-in coloca, em sua saída padrão, o prefixo "OK –" seguido da lista de máquinas virtuais. Caso o hash não possua este uuid cadastrado, significa que a máquina física não hospeda máquinas virtuais, e o plug-in coloca em sua saída "Warning – None" para indicar isso. Para o Nagios, uma informação importante é o código de retorno de um plug-in. Este código indica a situação do serviço que para o Nagios pode ser "OK", "Warning", "Critical" ou "Unknown". O código de retorno deve ser obtido de uma estrutura hash fornecida pelo Nagios chamada "ERRORS", a qual deve ser importada do arquivo utils.pm.

Se alguma das estratégias passivas estiver sendo utilizada, o plug-in necessita montar um comando para informar passivamente a lista de máquinas virtuais ao Nagios. Para checagem passiva, o plug-in precisará montar apenas um comando para a máquina física onde o plug-in está sendo executado. Para checagem passiva agressiva, o plug-in deverá montar um comando para cada máquina física que está no cluster.

Para montar este comando, é necessário conhecer onde o programa "send_nsca" está instalado, o endereço IP do servidor Nagios, escolher um delimitador para os campos de informação de estado do serviço e o caminho completo para o arquivo de configuração do "send_nsca". O programa "send_nsca" espera uma string separada pelo delimitador escolhido, com o seguinte formato: "<nome_da_máquina>;<nome_do_serviço>;<estado_do_serviço>;<saída_do_serviço>". Nestes campos, serão colocadas, respectivamente, as seguintes informações: nome da máquina física, "virtual_machines", 0 para indicar que o serviço está "OK" e "OK –" seguido da lista de máquinas virtuais que esta máquina física hospeda. Esta lista é obtida das estruturas hash utilizando os mesmos passos descritos para a estratégia ativa.

O plug-in apenas irá disparar este comando ao servidor Nagios se este comando se diferenciar do informado anteriormente ou se o último comando foi disparado há mais de 10 minutos. Se o plug-in dispara o comando, gerado no passo anterior, ele grava este comando em um arquivo no diretório "/tmp" com o nome "nagios_plugin_<nome_da_máquina>". Logo, para descobrir se um comando é diferente do anterior, basta comparar o comando gerado com aquele armazenado neste arquivo. Além disso, para controlar há quanto tempo o último comando foi informado ao Nagios, o plug-in captura a informação de data/hora de criação deste arquivo e compara se a diferença entre a data/hora atual e a data/hora de criação é superior há 10 minutos.

Este plug-in necessita executar o comando "xe" disponibilizado pelo XenServer. Este comando só pode ser executado pelo usuário "root". Os plug-ins que são chamados pelo NRPE executam com o usuário do NRPE chamado "nrpe". Para contornar este problema, o usuário "nrpe" precisa receber permissão para executar o comando "sudo" para, então, executar o plug-in "check_xen_virtual_machines" como usuário "root".

4.2 Plug-in Check_physical para o Nagios

Este plug-in foi desenvolvido em Perl pelos mesmos motivos do plug-in anterior. Além disso, como este plug-in é executado no servidor Nagios, onde o processo do Nagios efetuará a chamada deste plug-in, ele pode ser executado no ePN (embedded Perl Nagios), que é um interpretador Perl incorporado ao Nagios. A vantagem em utilizar o ePN é que o interpretador já está instanciado aguardando para interpretar o

plug-in. Caso outra linguagem fosse utilizada, o Nagios deveria criar um novo processo para instanciar o interpretador de outra linguagem ou o programa executável do plug-in.

Este plug-in aceita três parâmetros: -H, -c e –a. Em –H, deve ser informado o nome da máquina virtual que se deseja obter informações da máquina física. Em –c, deve ser informado o comando de checagem que será executado na máquina física. E em –a, devem estar os argumentos para o comando de checagem que será executado na máquina física.

Com o nome da máquina virtual informado nos parâmetros, o plug-in consegue determinar qual o nome do arquivo, gravado pelo módulo NEB, que possui o nome da máquina física que ele deve contatar. Com esta informação, o plug-in monta uma chamada ao plug-in "check_nrpe", repassando o comando de checagem obtido como argumento e os parâmetros para este comando de checagem, também obtidos como argumento. A saída deste plug-in é desviada para um arquivo em "/tmp" com o nome "<nome_da_maquina>_<comando_de_checagem>". Isto foi necessário porque o plug-in necessita capturar o código de retorno do plug-in "check_nrpe" para repassar ao Nagios. Para chamadas de sistema em Perl, só é possível capturar a saída ou o código de retorno da chamada de sistema.

O plug-in captura a saída do "check_nrpe", a qual foi desviada para o arquivo em "/tmp", e o código de retorno capturado da chamada do plug-in. Note que nesta saída estará o estado do serviço que foi monitorado na máquina física. O plug-in coloca em sua saída padrão o estado deste serviço e termina sua execução com o mesmo código de retorno do plug-in "check_nrpe".

4.3 Módulo Detect_host_change para o Nagios Event Broker

O Nagios é desenvolvido utilizando-se a linguagem de programação C. Por este motivo, o módulo "detect_host_change" foi desenvolvido nesta linguagem.

Este módulo escuta o canal "SERVICE_STATE_CHANGE" do NEB. Quando este módulo é chamado pelo processo do Nagios, ele faz uma série de testes para descobrir se o evento que o Nagios está informando é de seu interesse. Ele testa se a estrutura de dados recebida está ok; se esta estrutura indica que o estado do serviço foi processado corretamente; se o estado do serviço informado é "0", indicando que o serviço está com estado "OK"; se a estrutura de dados aponta para uma estrutura de dados de serviço correta; se o serviço cujo evento foi recebido chama-se "virtual_machines". Por fim, se todas estas condições forem satisfeitas, ele continua seu processamento. Em seguida, ele prepara uma estrutura de dados que conterá os parâmetros passados para um thread e dispara este thread. Depois de disparar o thread, ele captura a saída do estado do serviço, onde a lista de máquinas virtuais se encontra e, para cada máquina virtual nesta lista, atualiza as estruturas de dados internas do Nagios para que a máquina física seja considerada pai da máquina virtual e a máquina virtual seja considerada filha da máquina física.

O thread disparado pelo módulo NEB executa a maior parte das tarefas pertinentes ao módulo. Logo no início, ele testa um semáforo que é utilizado para que exista apenas um thread atualizando as estruturas de dados e os arquivos do módulo e do Nagios de cada vez. Depois de obter acesso, ele verifica, na estrutura de dados, se alguma máquina virtual, que era hospedada anteriormente por esta máquina física, não está na lista informada pela máquina física. Isto é feito para garantir que informações órfãs não

permaneçam na estrutura de dados. Se isto ocorrer, uma variável é ajustada para indicar que houve mudança no relacionamento entre máquinas virtuais e físicas. Depois, a thread insere, uma a uma, as máquinas virtuais nesta estrutura de dados. A função de inserção verifica se a informação que está para ser inserida é igual à armazenada. Se ela for idêntica apenas retorna. Se não for, atualiza a estrutura de dados e ajusta uma variável indicando que houve mudança no relacionamento entre as máquinas virtuais e físicas. Se a variável que indica mudança no relacionamento foi ajustada, o thread remove todos os arquivos do diretório que indicam onde cada máquina virtual está sendo executada, recria os arquivos com base nas novas informações constantes da estrutura de dados e atualiza o arquivo "object_cache_file" para que estas mudanças sejam colocadas na interface do Nagios.

A tarefa de atualizar o arquivo "object_cache_file" é a mais pesada desenvolvida pelo módulo e obrigou a utilização de threads. Se threads não fossem utilizados, o processo do Nagios ficaria parado esperando que o módulo terminasse de processar este arquivo. O que certamente levaria a problemas de desempenho no sistema Nagios inteiro.

Neste módulo, estão sendo utilizados threads no nível de sistema e do tipo "DETACHED". São deste tipo porque nunca são sincronizadas com outros threads. Eles fazem sua tarefa e terminam. Durante o desenvolvimento, antes de torná-las deste tipo, houve problemas quanto ao número máximo de threads de um processo. Quando threads são do tipo "JOINABLE" eles permanecem ativos para o processo que os disparou, porque outros threads podem necessitar sincronizar-se com eles. Logo, o número máximo de threads para um processo é alcançado e impede que novos threads sejam criados. Os threads que já processaram devem ser retirados do processo. Para facilitar este processo, elas já são criadas independentes do processo do módulo. Isto é feito criando-os do tipo "DETACHED".

O módulo não faz alterações diretamente sobre o arquivo "object_cache_file", a fim de minimizar problemas de concorrência no acesso a este arquivo. Imagine que, enquanto o módulo faz alterações neste arquivo, a interface do Nagios está processando este arquivo para buscar informações sobre as máquinas e serviços monitorados. O módulo faz uma cópia do arquivo "object_cache_file". Este arquivo, chamado "object_cache_file.saved", é processado pelo módulo toda vez. Ou seja, as alterações são sempre feitas em uma cópia do arquivo "object_cache_file" original gerado pelo Nagios. Para diferenciar arquivos gerados pelo Nagios daqueles manipulados pelo módulo, o módulo grava – na primeira linha do arquivo – a seguinte frase "# Patched by detect_host_change". Assim, toda vez que o módulo tenta atualizar o arquivo para a interface, ele testa se o arquivo "object_cache_file" possui esta frase na primeira linha. Se não possuir, ele copia este arquivo para "object_cache_file.saved". Se possuir a linha, o módulo utiliza o arquivo "object_cache_file.saved" que já possui. O módulo copia, linha a linha, o arquivo "object_cache_file.saved" fazendo as alterações devidas e criando um arquivo "object_cache_file.new". Durante esta etapa, o arquivo "object_cache_file" continua disponível para a interface do Nagios. Ao final das modificações, o arquivo "object_cache_file.new" é renomeado para "object_cache_file". Deve-se notar que o processamento do arquivo leva muito mais tempo que a simples renomeação do arquivo e isto minimiza a concorrência neste arquivo.

Outro problema é quando uma máquina virtual é parada e não é mais informada por nenhuma máquina física. O arquivo "object_cache_file" conterá, na definição desta

máquina virtual, como sendo pai a máquina física que a executava antes dela ser parada. Neste caso, a máquina virtual será retirada das estruturas de dados e, se o arquivo "object_cache_file" fosse processado novamente, esta máquina não seria detectada como sendo uma máquina virtual do cluster, por não estar na estrutura de dados do módulo. Logo, o módulo deveria manter uma lista de máquinas virtuais que já passaram pelo módulo. Quando o módulo faz alterações em uma cópia de um arquivo "object_cache_file" original, este problema não acontece porque, para o Nagios, aquela máquina virtual não possui parentesco, ou possui um parentesco informado pelo administrador. Logo, quando esta máquina é parada, a informação de parentesco que permanece é aquela que foi configurada no Nagios pelo administrador.

5 AVALIAÇÃO

A avaliação da solução irá preocupar-se em três aspectos principais: tempo necessário para que a interface do Nagios mostre a informação correta, utilização de rede em cada estratégia e teste de escalabilidade do sistema. A seguir, serão apresentados o ambiente monitorado, os resultados esperados e os resultados obtidos para cada um dos testes.

5.1 Ambiente Monitorado

O cluster Citrix XenServer possui cinco máquinas físicas que usam dois processadores Intel Xeon QuadCore E5430 com 12 MB de cache L2 e 16 GB de memória principal. O servidor de armazenamento possui, aproximadamente, 1 TB de espaço para armazenamento de máquinas virtuais. O servidor Nagios utiliza um processador Intel Core2 Duo E8400 com 6 MB de cache L2 e 2 GB de memória principal. Todas as conexões de rede são Gigabit Ethernet. A Figura 5.1 mostra o esquema de rede do ambiente de experimentos.

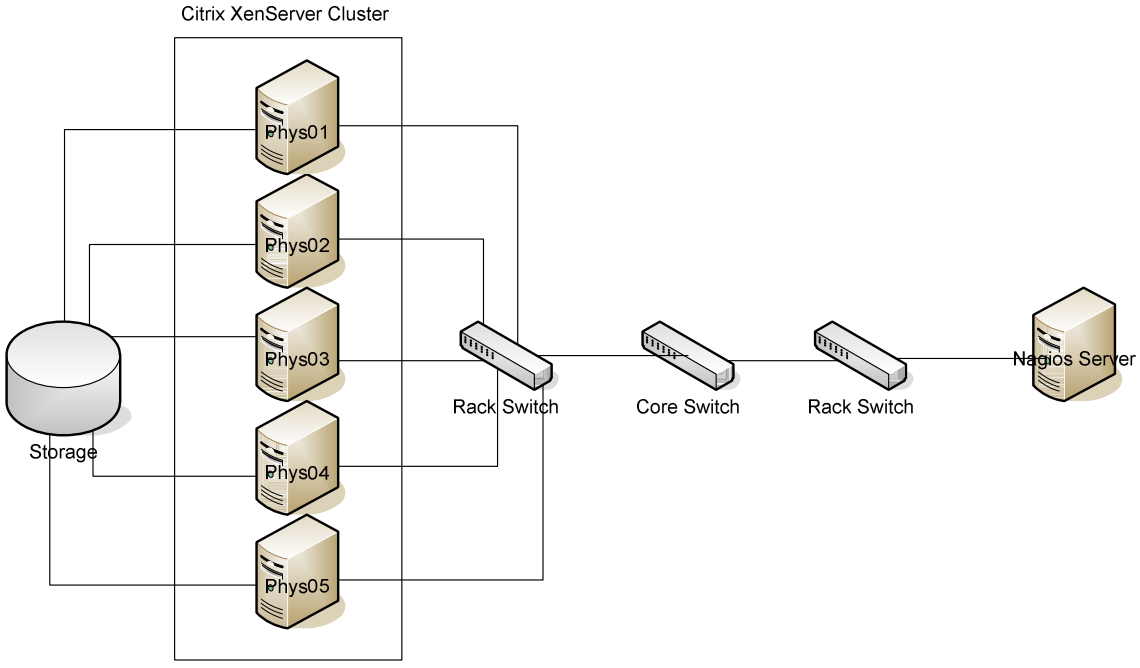

Figura 5.1: Ambiente de rede monitorado durante os experimentos.

Neste ambiente, estão instaladas, aproximadamente, 20 máquinas virtuais prestando os mais diversos serviços, tais como Domain Controller (DC), Ambiente Virtual de Aprendizado (AVA), webmail, ambiente de colaboração, base de conhecimento,

plataforma para gerência de projetos e servidores de aplicação Java e “Ruby On Rails”. Mesmo nesse ambiente configurado para alta disponibilidade, já ocorreram problemas de disponibilidade de alguns destes servidores. Por exemplo, uma máquina física do cluster apresentou um problema físico e acabou desligando-se do cluster. Entretanto, as máquinas virtuais que estavam nesta máquina física não foram migradas e permaneceram fora do ar até que os administradores detectassem esta falha e as inicializassem em outra máquina física. Posteriormente, foi verificado um problema de configuração no cluster que impedia que as máquinas virtuais fossem migradas. Ou seja, mesmo em um ambiente dito de alta disponibilidade, podem ocorrer problemas de configuração que prejudiquem esta disponibilidade. Este fato deve chamar a atenção da necessidade de monitoramento mesmo em ambientes ditos confiáveis.

No caso, esta falha poderia ser detectada por uma ferramenta de monitoramento que verificasse se determinada máquina está no ar. Entretanto, se o problema fosse quanto à escassez de recursos em determinada máquina física do cluster, esta ferramenta não detectaria o problema. Se existirem problemas que impeçam a migração de máquinas virtuais, como ocorreu anteriormente, uma ferramenta de monitoramento que apenas verifica se a máquina está no ar não detectaria a falha. Pois, de fato a máquina virtual estará no ar, mas está enfrentando problemas de desempenho impostos pela concorrência pelos recursos da máquina física.

Uma solução como a que está sendo proposta detectaria este problema porque, juntamente com as informações da máquina virtual, se encontram as informações da máquina física. Por exemplo, o administrador receberia uma notificação do Nagios informando que determinada máquina virtual está com seu serviço “physical_load” ou “physical_memory” em estado crítico. Basta o administrador verificar o que está sobrecarregando a máquina física para resolver o problema.

5.2 Comparação de estratégias por tempo

O experimento consiste em promover a migração de uma máquina virtual e medir o tempo necessário para que a interface do Nagios mostre esta mudança. Este experimento foi efetuado com diferentes valores de intervalo de checagem. Um cenário configura o intervalo de checagem para 1 minuto, que é o menor tempo aceito pelo Nagios, e outro cenário utiliza 10 minutos, que é o tempo de checagem padrão do Nagios. Este experimento foi executado em um ambiente real, como descrito na Figura 5.1.

Este experimento foi executado nestes dois cenários para as três estratégias apresentadas. Para a estratégia utilizando checagem ativa, os intervalos de checagem foram configurados no servidor Nagios e este intervalo é controlado pelo processo do Nagios. Para as estratégias utilizando checagem passiva, os intervalos são configurados e controlados pela Cron do sistema.

Para que os valores médios medidos tenham uma validade estatística, o experimento foi repetido 30 vezes para cada cenário em cada estratégia. Para que estes valores não ficassem viciados, ao final da obtenção do valor de cada amostra, foi inserido um tempo aleatório da ordem de grandeza do intervalo de checagem utilizado.

Uma estimativa teórica pode ser feita para estas estratégias. A primeira estimativa a ser feita é o tempo máximo que a interface do Nagios levará para mostrar corretamente

as informações de relacionamento para cada estratégia. Para a estratégia de checagem ativa pode-se utilizar (1).

$$\text{Tempo máximo} = I + S + P + R + MP + RI \quad (1)$$

Em (1), I é o intervalo de checagem. S é o tempo necessário para enviar um comando de checagem para a máquina monitorada. P é o tempo de processamento do comando de checagem na máquina remota. R é o tempo necessário para receber a saída do comando de checagem. MP é o tempo de processamento do módulo NEB. E RI é o tempo de atualização da interface Web do Nagios. Todas estas variáveis devem ser fornecidas em segundos e o resultado será obtido em segundos.

Para S, P, R e MP serão assumidos o tempo de 1 segundo, que é uma estimativa pessimista. RI é 90 segundos, tempo padrão no Nagios, e I será alterado para os cenários de 1 minuto e 10 minutos. Assim, obtêm-se como tempos máximos 154 segundos e 694 segundos para estes cenários, respectivamente.

A mesma estimativa pode ser feita para as estratégias que utilizam checagem passiva. Neste caso, deve-se utilizar (2).

$$\text{Tempo Máximo} = I + P + R + E + MP + RI \quad (2)$$

Em (2), I é o intervalo de checagem. P é o tempo de processamento do comando de checagem. R é o tempo necessário para o servidor Nagios receber a saída do comando de checagem. E é o intervalo de tempo em que o Nagios lê os comandos no “External Command File”. MP é o tempo de processamento do módulo NEB. E RI é o tempo de atualização da interface Web do Nagios. Todas estas variáveis devem ser fornecidas em segundos e o resultado será obtido em segundos.

Para P, R e MP será assumido o tempo de 1 segundo, que é uma estimativa pessimista. RI é de 90 segundos. E é de 15 segundos e, modificando I para os cenários de 1 minuto e 10 minutos, obtém-se 168 segundos e 708 segundos para estas estratégias, respectivamente. A estimativa de tempo máximo para a estratégia de checagem passiva agressiva é a mesma, pois os componentes são os mesmos e os aprimoramentos feitos para a estratégia agressiva não influenciam no tempo máximo.

Outra estimativa interessante é o tempo médio que é esperado no experimento para cada estratégia. Para obter esta estimativa, serão considerados os tempos de intervalo como sendo a metade do tempo de valor nominal. Esta estimativa leva em conta o tempo, em média, que ocorrerão estes intervalos. Isto será feito para o intervalo de checagem, atualização da interface e leitura do “External Command File”. As equações modificadas são mostradas em (3) e (4) para as estratégias ativa e passiva, respectivamente.

$$\text{Tempo Médio} = I/2 + S + P + R + MP + RI/2 \quad (3)$$

$$\text{Tempo Médio} = I/2 + P + R + E/2 + MP + RI/2 \quad (4)$$

A estratégia utilizando checagem passiva agressiva tem uma pequena modificação que minimiza o seu tempo médio. Como a primeira máquina física que detectar uma

mudança no relacionamento entre as máquinas físicas e virtuais irá informar esta mudança para o processo Nagios, será considerado como tempo médio do intervalo de checagem o seu valor nominal pela metade, como nas estratégias anteriores, dividido pelo número de máquinas físicas no cluster. Esta estimativa é mostrada em (5).

$$\text{Tempo Médio} = (I/2)/N + P + R + E/2 + MP + RI/2 \quad (5)$$

Em (5), N é o número de máquinas físicas no cluster Citrix XenServer.

Para a estratégia utilizando checagem ativa, o tempo médio com I de 1 minuto é 79 segundos e com I de 10 minutos é de 349 segundos. Para estratégia utilizando checagem passiva, o tempo médio com I de 1 minuto é 85,5 segundos e com I de 10 minutos é 355,5 segundos. Para estratégia utilizando checagem passiva agressiva com N de cinco servidores, o tempo médio com I de 1 minuto é 61,5 segundos e com I de 10 minutos é 115,5 segundos.

A Tabela 5.1 mostra os tempos estimados e medidos em 30 experimentos para cada cenário utilizando a estratégia de checagem ativa.

Tabela 5.1: Tempos medidos e estimados utilizando checagem ativa

Medida	*Intervalo de checagem*	
	1 minuto	*10 minutos*
Tempo máximo estimado	154 s	694 s
Tempo máximo medido	130 s	659 s
Tempo médio estimado	79 s	349 s
Tempo médio medido	66,67 s	349,97 s

A Tabela 5.2 mostra os tempos estimados e medidos em 30 experimentos para cada cenário utilizando a estratégia de checagem passiva.

Tabela 5.2: Tempos medidos e estimados utilizando checagem passiva

Medida	*Intervalo de checagem*	
	1 minuto	*10 minutos*
Tempo máximo estimado	168 s	708 s
Tempo máximo medido	149 s	574 s
Tempo médio estimado	85,5 s	355,5 s
Tempo médio medido	77,5 s	377,54 s

A Tabela 5.3 mostra os tempos estimados e medidos em 30 experimentos para cada cenário utilizando a estratégia de checagem passiva agressiva.

Tabela 5.3: Tempos medidos e estimados utilizando checagem passiva agressiva

Medida	*Intervalo de checagem*	
	1 minuto	*10 minutos*
Tempo máximo estimado	168 s	708 s
Tempo máximo medido	147 s	557 s
Tempo médio estimado	61,5 s	115,5 s
Tempo médio medido	78,37 s	401,27 s

A estratégia de checagem ativa é a mais rápida para os cenários com intervalo de checagem de 1 minuto e de 10 minutos. As estratégias que utilizam checagem passiva dependem de um tempo adicional que o Nagios leva para ler o "External Command File" e, por este motivo, acabam sendo mais lentas.

A estratégia utilizando checagem passiva agressiva não alcança seu objetivo. Esta estratégia tira vantagem do não determinismo do momento em que o comando de checagem irá ser chamado. Este comando é chamado pela Cron do sistema que testa quais os comandos que devem ser executados a cada minuto do relógio local da máquina física. Como os relógios locais das máquinas físicas do cluster estão sincronizados as Crons de sistema de todas as máquinas físicas chamam o script ao mesmo tempo. Isto torna determinístico o momento da execução do comando e acaba com a vantagem desta estratégia. Isto pode ser comprovado comparando-se os tempos médios das estratégias utilizando checagem passiva e checagem passiva agressiva, as quais são semelhantes.

5.3 Comparação de estratégias por utilização de rede

Este experimento consiste em capturar todo o tráfego gerado por cada estratégia. Para que uma comparação entre estratégias com diferentes valores de intervalo de checagem seja possível, esta comparação será feita considerando o tráfego gerado em uma hora. As duas estratégias baseadas em checagem passiva podem minimizar a utilização de rede se detectar que não ocorreu uma mudança na informação de relacionamento entre as máquinas virtuais e físicas desde a última execução do comando de checagem. Por outro lado, a cada 10 minutos, estes scripts forçam o envio desta informação ao Nagios. Isto é necessário no caso de uma reinicialização do processo do Nagios. Sem esta checagem forçada, a máquina física apenas enviaria suas informações se uma máquina virtual migrasse de/para ela. Com essa informação, pode-se delimitar o número máximo e mínimo de transmissões em cada estratégia a partir de uma máquina física. Na estratégia utilizando checagem passiva agressiva, está sendo considerado que cinco máquinas físicas fazem parte do cluster. Nesta estratégia, as informações de relacionamento entre máquinas virtuais e físicas de todas as máquinas físicas do cluster são informadas por cada máquina física, o que aumenta o número de transmissões. A Tabela 5.4 mostra o número de transmissões efetuadas por uma máquina física em uma hora.

Tabela 5.4: Números mínimos e máximos de transmissões em uma hora

Estratégia	*Intervalo de Checagem*	*Número de Transmissões*	
		Mínimo	*Máximo*
Checagem ativa	1 minuto	60	60
	10 minutos	6	6
Checagem passiva	1 minuto	6	60
	10 minutos	6	6
Checagem passiva agressiva	1 minuto	30	300
	10 minutos	30	30

No experimento, foram capturadas transmissões TCP para cada estratégia e obtido o tamanho médio destas transmissões para checagem ativa e passiva, que são 1960 bytes e 1594 bytes, respectivamente. A Tabela 5.5 mostra o tráfego acumulado em uma hora.

Tabela 5.5: Números mínimos e máximos de bytes transmitidos em uma hora

Estratégia	*Intervalo de Checagem*	*Tamanho das Transmissões*	
		Mínimo	*Máximo*
Checagem ativa	1 minuto	117.600	117.600
	10 minutos	11.760	11.760
Checagem passiva	1 minuto	9.564	95.640
	10 minutos	9.564	9.564
Checagem passiva agressiva	1 minuto	47.820	478.200
	10 minutos	47.820	47.820

A estratégia utilizando checagem passiva é a mais eficiente na utilização de rede. Esta estratégia tira vantagem no tamanho individual de suas transmissões e de seu aprimoramento que minimiza o número de transmissões. A estratégia utilizando checagem ativa é melhor que a checagem passiva agressiva, exceto no cenário com 1 minuto de intervalo de checagem e em ambientes onde raramente ocorre uma mudança nas informações de relacionamento.

5.4 Teste de Escalabilidade

O objetivo deste experimento é avaliar o comportamento dos plug-ins e módulos em sistemas maiores. Os testes anteriores foram executados em ambiente real, mas este foi executado artificialmente utilizando uma estratégia de checagem passiva. O Nagios foi configurado para quatro cenários: 25 máquinas físicas e 125 máquinas virtuais, 50 máquinas físicas e 250 máquinas virtuais, 75 máquinas físicas e 375 máquinas virtuais e 100 máquinas físicas e 500 máquinas virtuais. Cada máquina física hospeda cinco máquinas virtuais.

Nota-se que é impossível para um humano perceber uma mudança nestes cenários. Por isso, o experimento consiste em reiniciar o processo Nagios para limpar toda a informação que ele tenha aprendido. Depois disso, a interface do Nagios mostra todas as máquinas virtuais e físicas lado a lado, em outras palavras, sem qualquer informação de relacionamento. O próximo passo é executar um comando que informa ao Nagios as listas de todas as máquinas virtuais de cada máquina física. O experimento termina quando a interface do Nagios mostra todas as máquinas virtuais associadas com suas respectivas máquinas físicas. Isto é simples de verificar visualmente porque basta visualizar que nenhuma máquina virtual pode estar sozinha na interface. O tempo que é medido é o tempo entre o início da execução do script, que informa as listas de máquinas virtuais de cada máquina física, até que a interface mostre todas máquinas virtuais associadas com sua máquina física.

Os resultados expostos na Figura 5.2 mostram que a solução escala com o aumento do ambiente monitorado. Entretanto, em cenários como 150 máquinas físicas com 750 máquinas virtuais, a interface do Nagios começa a mostrar um comportamento intermitente. A interface do Nagios muitas vezes não funciona neste cenário.

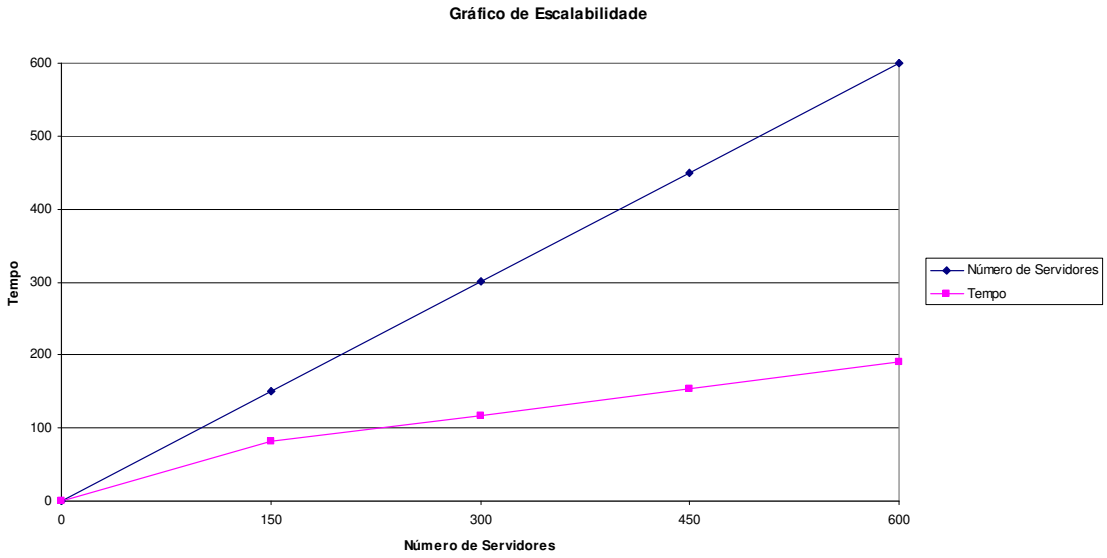

Figura 5.2: Gráfico de escalabilidade.

6 CONCLUSÃO

Com as modificações feitas no Nagios para incorporar informações pertinentes à virtualização um administrador, pode colocar juntas as informações de estado dos serviços de cada máquina virtual com as métricas dos recursos coletadas da máquina física que a hospeda. Isto é feito dinamicamente para serem sensíveis às migrações destas máquinas virtuais sem qualquer intervenção do administrador. A interface Web do Nagios mostra graficamente as informações de relacionamento entre as máquinas físicas e virtuais no seu Mapa. Neste mapa, as máquinas físicas são mostradas como pais de cada máquina virtual que hospeda.

Dentre as estratégias apresentadas, aquela que apresentou menor tempo de latência para mostrar corretamente as informações de relacionamento foi a estratégia que utiliza checagem ativa. Para as mesmas estratégias, aquela que apresentou menor utilização de rede foi a estratégia que utiliza checagem passiva. Logo, se um administrador não possui problemas de escassez de recursos de rede, recomenda-se a utilização de checagem ativa. Caso tenha problemas de falta de recursos de rede, recomenda-se a utilização de checagem passiva. A checagem passiva agressiva não deve ser utilizada por consumir mais recursos de rede e apresentar um tempo de resposta semelhante ao tempo da checagem passiva, a qual é mais eficiente na utilização de rede.

O único ambiente que possibilitaria um uso melhor para checagem passiva agressiva seria àquele onde os relógios das máquinas físicas não estão sincronizados. Desta forma, o não determinismo existente no momento em que o comando de checagem será executado pode trazer benefícios. No caso do Citrix XenServer, os relógios das máquinas físicas estão sincronizados, impossibilitando a utilização desta estratégia.

A estratégia que utiliza checagem ativa é melhor gerenciável porque o intervalo de checagem é controlado pelo Nagios. Neste caso, em uma reconfiguração que envolva modificar o intervalo de checagem, basta alterar o “service template” que todos os serviços baseados neste “service template” terão seu intervalo de checagem alterado. Para estratégia utilizando checagem passiva, o administrador deverá alterar manualmente o Cron de sistema de cada uma das máquinas físicas para modificar o intervalo de checagem. Dessa maneira, corre-se o risco de ter configurações distintas em cada máquina física que façam com que o monitoramento se comporte de maneira diferente, dependendo das máquinas envolvidas.

O teste de escalabilidade mostrou que esta solução pode ser empregada em sistemas de grande porte. Entretanto, o Nagios teve comportamento inesperado em ambientes com 900 servidores (250 máquinas físicas e 750 máquinas virtuais). Neste caso, o administrador deverá utilizar dois ou mais servidores Nagios e empregar estes componentes em cada servidor. Ou então, melhorar a configuração do servidor Nagios nos recursos que comprometeram sua utilização nestes ambientes maiores.

Este trabalho foi baseado no ambiente de virtualização Citrix XenServer, mas pode ser facilmente adaptado para outros ambientes. O único elemento da solução que é específico para esta plataforma é o plug-in “check_xen_virtual_machines”. O módulo NEB, que faz o trabalho pesado na solução, espera uma lista de máquinas virtuais para cada máquina física como “OK – virtual01,virtual02,virtual03”. Qualquer um pode desenvolver um plug-in que informa ao Nagios esta lista para o seu ambiente de virtualização.

Outra melhoria para monitorar corretamente as máquinas físicas do Citrix XenServer é o desenvolvimento de plug-ins para o Nagios que coletem informações de estado dos recursos do Hypervisor do XenServer. Os plug-ins fornecidos pela comunidade do Nagios funcionam apenas com o Hypervisor do Xen de código aberto e devem ser adaptados para funcionar com o Citrix XenServer. Ou o administrador pode utilizar o plug-in “check_snmp” para coletar estas métricas da interface SNMP da máquina física.

REFERÊNCIAS

BAIARDI, F.; SGANDURRA, D. Building Trustworthy Intrusion Detection through VM Introspection. In: INTERNATIONAL SYMPOSIUM ON INFORMATION ASSURANCE AND SECURITY, 3., 2007, Manchester. **Proceedings…** [S.l.:s.n], 2007. p. 209-214.

CARISSIMI, A. S. Virtualização: princípios básicos e aplicações. In: ESCOLA REGIONAL DE ALTO DESEMPENHO, 9., 2009, Caxias do Sul. **Anais...** Porto Alegre: SBC, 2009. p. 39-69.

FRASER, T.; EVENSON, M. R.; ARBAUGH, W. A. VICI-Virtual Machine Instrospection for Cognitive Immunity. In: ANNUAL COMPUTER SECURITY APPLICATIONS CONFERENCE, 2008, Anaheim CA. **Proceedings…** [S.l.:s.n], 2008. p. 87-96.

HUANG, H. et al. Building end-to-end Management Analytics for Enterprise Data Centers. In: INTERNATIONAL SYMPOSIUM ON INTEGRATED NETWORK MANAGEMENT, 2009, Long Island NY. **Proceedings…** [S.l.:s.n], 2009. p. 661-675.

MACHIDA, F.; KAWATO, M.; MAENO, Y. Adaptative Monitoring for Virtual Machine Based Reconfigurable Enterprise Systems. In: INTERNATIONAL CONFERENCE ON AUTONOMIC AND AUTONOMOUS SYSTEMS, 3., 2007, Atenas. **Proceedings…** [S.l.:s.n], 2007. p. 8.

PAYNE, B. D.; CARBONE, M. D. P. de A.; LEE, W. Secure and Flexible Monitoring of Virtual Machines, In: ANNUAL COMPUTER SECURITY APPLICATIONS CONFERENCE, 23., 2007, Miami Beach, FL. **Proceedings…** [S.l.:s.n], 2007. p. 385-397.

SHAO, Z.; JIN, H.; LU, X.. PMonitor: a lighweight performance monitor for virtual machines. In: INTERNATIONAL WORKSHOP ON EDUCATION TECHNOLOGY AND COMPUTER SCIENCE, 1., 2009, Wuhan, Hubei. **Proceedings…** [S.l.:s.n], 2009. p. 689-693.

SMITH, J. E.; NAIR, R. Process Virtual Machines; System Virtual Machines. In: SMITH, J. E.; NAIR, R. **Virtual Machines: versatile platforms for systems and processes.** Amsterdam: Elsevier, 2005. p. 83-145; 369-442.

VERMA, D. C. Monitoring. In: VERMA, D. C. **Principles of Computer Systems and Network Management.** Nova Iorque: Springer, 2009. p. 111-134.