Pontifícia Universidade Católica do Rio Grande do Sul
Faculdade de Informática
Programa de Pós-Graduação em Ciência da Computação

Uma Arquitetura Baseada em WBEM para o Gerenciamento de um *Cluster* de Máquinas Virtuais

Everton Batista Petró Alexandre

Dissertação apresentada como requisito parcial à obtenção do grau de mestre em Ciência da Computação

Orientador: Prof. Dr. César Augusto F. De Rose

Porto Alegre
2011

Pontifícia Universidade Católica do Rio Grande do Sul
FACULDADE DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

TERMO DE APRESENTAÇÃO DE DISSERTAÇÃO DE MESTRADO

Dissertação intitulada **"Uma Arquitetura Baseada em WBEM para o Gerenciamento de um *Cluster* de Máquinas Virtuais"**, apresentada por Everton Batista Petró Alexandre, como parte dos requisitos para obtenção do grau de Mestre em Ciência da Computação, Processamento Paralelo e Distribuído , aprovada em 30/03/09 pela Comissão Examinadora:

Prof. Dr. César Augusto Fonticielha De Rose – Orientador — PPGCC/PUCRS

Prof. Dr. Avelino Francisco Zorzo - — PPGCC/PUCRS

Prof. Dr. Antônio Marinho Pilla Barcellos – — UFRGS

Homologada em...../...../....., conforme Ata No. ..... pela Comissão Coordenadora.

Prof. Dr. Fernando Gehm Moraes
Coordenador.

PUCRS
Campus Central
Av. Ipiranga, 6681 – P32 – sala 507 – CEP: 90619-900
Fone: (51) 3320-3611 – Fax (51) 3320-3621
E-mail: ppgcc@pucrs.br
www.pucrs.br/facin/pos

Dados Internacionais de Catalogação na Publicação (CIP)

A381a Alexandre, Everton Batista Petró
Uma arquitetura baseada em WBEM para o gerenciamento de um *cluster* de máquinas virtuais / Everton Batista Petró Alexandre. – Porto Alegre, 2011.
72 p.

Diss. (Mestrado) – Fac. de Informática, PUCRS.
Orientador: Prof. Dr. César Augusto Fonticielha De Rose.

1. Informática. 2. Sistemas Distribuídos. 3. Máquinas Virtuais. 4. Rede de Computadores – Gêrencia. 5. Cluster – Informática. I. De Rose, César Augusto Fonticielha. II. Título.

CDD 004.36

Ficha Catalográfica elaborada pelo
Setor de Tratamento da Informação da BC-PUCRS

À minha família

Agradecimentos

Ao professor César Augusto F. De Rose.

Ao Rodrigo Neves Calheiros.

Aos colegas de CPPH, Andriele Busatto do Carmo, Yves Shiga e Roberto Karpinski.

À Pontifícia Universidade Católica do Rio Grande do Sul e à Hewlett-Packard pela concessão da bolsa de estudos.

Uma Arquitetura Baseada em WBEM para o Gerenciamento de um *Cluster* de Máquinas Virtuais

Resumo

Diversas pesquisas têm utilizado *clusters* de máquinas virtuais para emulação de sistemas distribuídos. No entanto, a virtualização cria um nível de indireção no ambiente. Para controlar um ambiente tradicional, que não usa a tecnologia de virtualização, é necessária apenas a referência para a máquina física. No entanto, para controlar um *cluster* de máquinas virtuais, necessita-se da referência para a máquina física e a referência para a máquina virtual. Outra dificuldade encontrada no gerenciamento de um *cluster* de máquinas virtuais está relacionada com a construção e configuração do ambiente virtual. O custo de gerenciar o ciclo de vida de cada máquina virtual do ambiente e as aplicações que executam sobre o *cluster* virtualizado de forma manual é grande. Desta forma, a arquitetura apresentada através deste trabalho tem por objetivo disponibilizar serviços para aplicações de gerência, permitindo a construção automatizada de *clusters* de máquinas virtuais, a monitoração destes ambientes e a gerência das aplicações executadas sobre o *cluster* virtualizado. Além disso, a arquitetura diminui a complexidade causada pela virtualização, pois com a utilização dos serviços disponibilizados, as aplicações de gerência passam a referenciar apenas as máquinas virtuais.

Palavras-chave: Gerenciamento de Recursos. Virtualização. Padrões de Gerenciamento de Sistemas. *Cluster* de Máquinas Virtuais. Emulação de Sistemas Distribuídos.

A WBEM-Based Architecture for Managing a Cluster of Virtual Machines

Abstract

A number of researches have been used clusters of virtual machines to emulate distributed systems. However, virtualization creates a level of indirection in the environment. In order to control a traditional environment that does not use the virtualization technology, only the reference for the physical machine is required. Nevertheless, in order to control a cluster of virtual machines, a reference is required both for the physical machine and for the virtual machine. Another difficulty found in the management of a cluster of virtual machines is related to the construction and setting up of the virtual environment. The cost related to manual management of virtual machines lifecycle and applications executing on the virtualized cluster of the environment is remarkable. Therefore, the architecture presented is intended to make available services for management applications, allowing automated construction of clusters of virtual machines, monitoring of such environments, and management of applications executed over the virtualized cluster. Additionally, the architecture decreases the complexity caused by the virtualization since, from the use of the services made available, the management applications start to reference the virtual machines only.

Keywords: Resources Management. Virtualization. Systems Management Standards. Cluster of Virtual Machines. Emulation of Distributed Systems.

Lista de Figuras

Lista de Tabelas

Lista de Siglas

CIM	*Common Information Model*
CIMOM	*Common Information Model Object Manager*
CPU	*Central Processing Unit*
DHCP	*Dynamic Host Configuration Protocol*
DMTF	*Distributed Management Task Force*
HTTP	*Hypertext Transfer Protocol*
IP	*Internet Protocol*
MIB	*Management Information Base*
MMV	*Monitor de Máquinas Virtuais*
MOF	*Managed Object Format*
MV	*Máquina Virtual*
RPC	*Remote Procedure Call*
SNMP	*Simple Network Management Protocol*
SMI	*Storage Management Initiative*
SMI-S	*Storage Management Initiative-Specification*
SNIA	*Storage Networking Industry Association*
SQL	*Structured Query Language*
TI	*Tecnologia da Informação*
TMN	*Telecommunications Management Network*
UA	*User Agents*
UML	*Unified Modeling Language*
WBEM	*Web-Based Enterprise Management*
WMI	*Windows Management Instrumentation*
XML	*Extensible Markup Language*

Sumário

1 Introdução

Entre as décadas de 60 e 90 os computadores passaram por significativas transformações. No início dos anos 60 os computadores existentes eram grandes e caros. O poder de processamento das máquinas existentes era reduzido. Tais características exigiam que a administração destes recursos fosse realizada de modo centralizado. A partir dos anos 80, o barateamento dos equipamentos e o significativo aumento do poder de processamento tornaram evidente a mudança de paradigmas, onde a computação distribuída substitui o conceito de computação centralizada [37].

Nesse contexto, os avanços observados nas redes de computadores e o conseqüente aumento do poder de processamento viabilizaram o uso de sistemas distribuídos. Desta forma, tanto as instituições acadêmicas quanto as corporativas começaram a desenvolver e utilizar este tipo de plataforma. Atualmente, devido à evolução deste quadro, a maioria dos serviços corporativos passaram a incorporar componentes distribuídos. As diversas aplicações que usam a Internet como meio de comunicação são exemplos de sistemas distribuídos [19]. No entanto, as técnicas para a realização de testes em aplicações que operam nesse tipo de plataforma não evoluíram com a mesma intensidade que a utilização destas aplicações [50].

Algumas características dos sistemas distribuídos dificultam a avaliação de experimentos executados nesse tipo de ambiente: por serem compostos por diversos nodos, e estes nodos podendo deixar o sistema de forma arbitrária (devido a uma falha, por exemplo), pelas condições momentâneas da rede de comunicação [72]. Estas características dificultam a reprodução de cenários de testes em sistemas distribuídos, já que dificilmente será possível obter a mesma configuração do sistema (o mesmo conjunto de nodos e as mesmas características de tráfego). Devido a estas limitações, muitas vezes a avaliação de aplicações não é realizada diretamente no ambiente distribuído.

Existem algumas técnicas para avaliar sistemas distribuídos, como a simulação, usada por Legrand [46] e a aplicação de métodos formais, usada por Plateau [57]. Outra técnica utilizada para a emulação de sistemas distribuídos é a virtualização. Nesse caso, utiliza-se um *cluster*, onde cada um de seus nodos hospedam uma ou mais máquinas virtuais (MVs). Estas máquinas virtuais, que atuarão como nodos do sistema distribuído emulado, são ligadas por uma rede virtual. Diversas pesquisas como [1, 11, 14, 42, 59] utilizam *clusters* virtualizados para testar e avaliar sistemas distribuídos.

No entanto, a gerência de *clusters* de máquinas virtuais é complexa. A virtualização cria um nível de indireção a mais entre o *hardware* e as aplicações de usuário [35]. Para controlar e obter informações de monitoração em um ambiente tradicional, sem a tecnologia de virtualiza-

ção, é necessária apenas a referência para uma máquina física. No entanto, para controlar um *cluster* de máquinas virtuais, necessita-se da referência para a máquina física e a referência para a máquina virtual. A Arquitetura de Gerenciamento em Ambientes Virtuais, proposta nesse trabalho, diminui o problema de indireção, causado pela tecnologia de virtualização. Considere, por exemplo, que se deseja monitorar a quantidade de memória utilizada por uma determinada máquina virtual. Para uma aplicação de gerência obter esta informação sem a utilização desta arquitetura, tal aplicação deveria primeiramente localizar o nodo que esta máquina virtual se encontra. Em seguida, a aplicação deveria invocar o comando da máquina virtual responsável por retornar a quantidade de memória usada.

Outra dificuldade encontrada no gerenciamento de um *cluster* de máquinas virtuais está relacionada com a construção e configuração do ambiente virtual. Um ambiente com muitas máquinas virtuais dificulta a configuração manual. A criação de máquinas virtuais, a execução de aplicações e a modificação de arquivos de configuração em cada máquina virtual do *cluster* são tarefas que se realizadas manualmente consomem um tempo considerável.

É objetivo deste trabalho minimizar os problemas apresentados anteriormente. Deste modo, este trabalho apresenta uma arquitetura de gerenciamento para um *cluster* de máquinas virtuais. Tal arquitetura diminui a complexidade causada pela virtualização, pois a aplicação de gerência passa a referenciar apenas as máquinas virtuais. É função da arquitetura verificar em qual nodo do *cluster* a máquina virtual está hospedada. Além disso, esta arquitetura provê serviços que permitem automatizar a construção e a configuração deste ambiente. Assim, serviços para a gerência do ciclo de vida das máquinas virtuais (criação, reinicialização, pausa, retomada, destruição), a inicialização e parada de aplicações que operam sobre as máquinas virtuais, a instalação de *softwares* e a alteração de arquivos de configuração foram criados para permitir automatização do ambiente. Desta forma, as aplicações de gerência utilizam estes serviços visando construir e alterar um ambiente virtual de forma automática.

Esta arquitetura abstrai ainda o ambiente distribuído, tornando transparente para a aplicação a localização das máquinas virtuais. Além disso, tal arquitetura abstrai a tecnologia de virtualização, visto que a aplicação interage com os serviços da arquitetura, não se relacionando diretamente com o virtualizador.

A comunicação entre os componentes da arquitetura é realizada por meio da especificação WBEM (*Web-Based Enterprise Management*) [39]. WBEM é uma especificação desenvolvida para unificar o gerenciamento de ambientes computacionais distribuídos. A especificação WBEM, criada pelo DMTF (*Distributed Management Task Force*) [25], provê um padrão para o controle e a monitoração de aplicações, serviços e dispositivos computacionais [33, 74]. Em suma, este padrão opera sobre o protocolo HTTP (*Hypertext Transfer Protocol*) e utiliza o modelo de dados CIM (*Common Information Model*) [26]. Clientes WBEM invocam operações CIM com objetivo de gerenciar o ambiente computacional distribuído. Os provedores WBEM atuam junto ao dispositivo gerenciado, recebendo invocações e retornando as informações solicitadas pelos clientes WBEM.

A especificação WBEM foi escolhida para elaboração da arquitetura, pois facilita o controle de aplicações. Existem tarefas que não são facilmente gerenciadas por outros protocolos, como o SNMP (*Simple Network Management Protocol*) [67]. Tal protocolo possui algumas limitações. As operações *get* e *set* do SNMP permitem apenas a busca e a alteração de informações de um determinado recurso. Estas operações não foram elaboradas para, por exemplo, invocar comandos imperativos, como reiniciar uma máquina virtual ou controlar uma aplicação. Já o padrão WBEM não dispõe somente de provedores de propriedade (*get* e *set* no SNMP). Além deste, existem os provedores de método, utilizados para chamada de métodos externos [39].

O restante do trabalho está estruturado da seguinte forma: o Capítulo 2 apresenta os conceitos relacionados a um *cluster* de máquinas virtuais. A seguir, no Capítulo 3, a gerência de ambientes computacionais é abordada. O Capítulo 4 apresenta a arquitetura desenvolvida nesse trabalho. No Capítulo 5, a validação e detalhes de implementação da arquitetura são apresentadas. Finalmente, no Capítulo 6, são apresentadas as considerações finais.

2 *Clusters* de Máquinas Virtuais

Como foi discutido no capítulo anterior, *clusters* de máquinas virtuais estão sendo utilizados para a emulação de sistemas distribuídos [1,11,14,42,59]. No entanto, existem outras utilidades. Considere, por exemplo, que diversos usuários compartilham um *cluster* de um laboratório de uma universidade. Estes usuários utilizam ambientes de execução distintos. Considere ainda que estes diversos ambientes de execução são incompatíveis. Uma das soluções para este caso é a utilização de *clusters* de máquinas virtuais. Neste caso, o usuário executa sua aplicação em um ambiente virtual, que possibilita a instalação de bibliotecas e *softwares* sem interferir no ambiente dos demais usuários [82].

Antes de definirmos um *cluster* virtual, convém apresentarmos alguns conceitos sobre *clusters* computacionais e máquinas virtuais. Desta forma, o presente capítulo está dividido nas seguintes seções: a Seção 2.1 aborda conceitos básicos sobre *clusters* computacionais. Na Seção 2.2, os principais conceitos relacionados com a tecnologia de virtualização são apresentados. Por fim, a Seção 2.3 aborda a definição de *clusters* de máquinas virtuais e suas possíveis utilizações.

2.1 *Clusters* Computacionais

Muito freqüentemente aplicações necessitam mais poder computacional do que um único computador pode prover. Um dos caminhos para contornar esta limitação é aumentar a capacidade dos componentes de um computador (aumento da velocidade de um processador, por exemplo). No entanto, as melhorias destes componentes estão limitadas pela velocidade da luz, leis termodinâmicas e elevados custos financeiros para a fabricação dos processadores [8]. Uma alternativa com um custo menor é conectar múltiplos computadores e coordenar seus esforços computacionais, agrupando o poder de processamento de diversos computadores para a resolução de um mesmo problema [9, 73]. Segundo Buyya [8], um *cluster* computacional é um tipo de sistema de processamento paralelo ou distribuído, que consiste em uma coleção de computadores autônomos interconectados. Tais computadores trabalham juntos, como um único recurso computacional.

Cada um destes computadores é normalmente chamado de nodo. Cada nodo possui um ou mais processadores, dispositivos de entrada e saída e memória próprios. Além disso, tais nodos são conectados através de uma rede de interconexão. Cada computador tem acesso apenas a

sua memória, com seus próprios espaços de endereçamento. Deste modo, a comunicação entre os nodos é realizada através de troca de mensagens [8]. A Figura 1 [8] apresenta a arquitetura típica de um *cluster*. Nesta figura, pode-se observar os diversos nodos comunicando-se por meio da rede de interconexão.

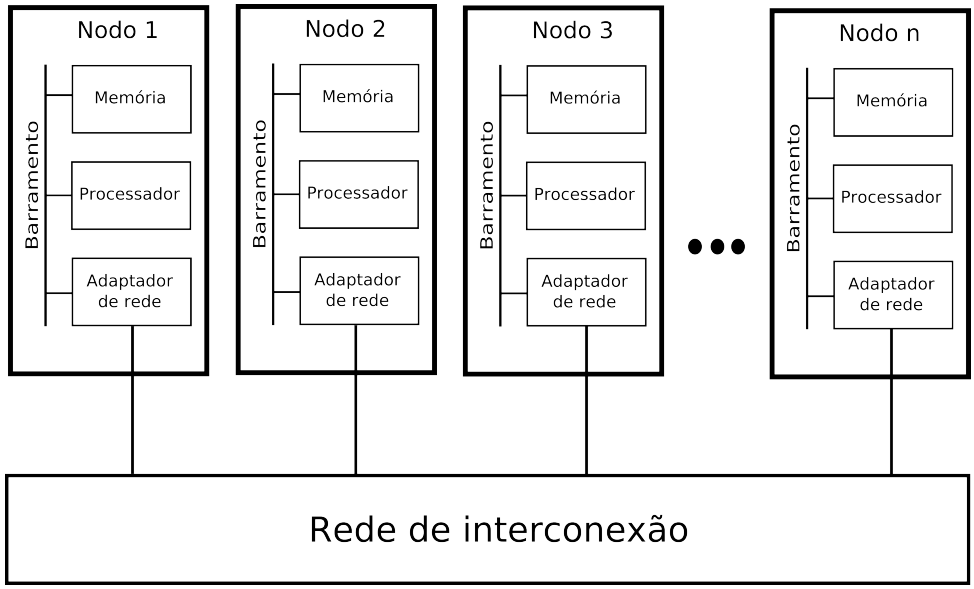

Figura 1: Arquitetura de um *cluster* computacional.

Clusters computacionais são projetados para execução de aplicações paralelas. Por esta razão, alguns fatores devem ser avaliados. Cada aplicação possui diferentes necessidades de velocidade de processamento e quantidade de memória RAM. Por isso, deve-se avaliar o número de máquinas a serem adquiridas, o número de processadores por nodo, a quantidade de memória RAM por nodo e o tamanho da memória *cache* dos processadores [78]. Além disso, a rede de interconexão também pode ser otimizada. Atualmente, existem duas tendências:

- *Clusters* conectados por redes de alta latência: são redes com um custo consideravelmente menor se comparadas com as redes de baixa latência. No entanto, são muito mais lentas para enviar uma mensagem de um nodo para outro do que as redes de baixa latência [20]. Este tipo de rede é adequado para aplicações que não tenham muita necessidade de comunicação. Um exemplo de redes de alta latência é a Ethernet.

- *Clusters* conectados por redes de baixa latência: as redes de baixa latência, ao contrário das redes de alta latência, são muito mais rápidas no envio de uma mensagem de um nodo para outro. No entanto, as placas de interconexão, desenvolvidas especificamente para máquinas agregadas, têm um custo mais alto do que as placas de redes de alta latência [20]. Portanto, este tipo de rede é usada por aplicações paralelas que tenham muita necessidade de comunicação. Como exemplos de redes de baixa latência, pode-se citar a Myrinet [6] e a Infiniband [4].

Os testes que validaram a arquitetura apresentada por meio do Capítulo 4 foram realizados através de um *cluster* conectado por uma rede de alta latência, visto que as aplicações utilizadas não necessitavam de comunicação entre os nodos.

2.2 Máquinas Virtuais

A virtualização de recursos computacionais está sendo utilizada com grande intensidade em pesquisas nas áreas da ciência da computação atualmente. No entanto, este conceito não é uma novidade. As primeiras máquinas virtuais foram desenvolvidas na década de 70 com o intuito de aproveitar os computadores da época de uma maneira mais eficiente. Assim, diversos usuários acessavam o *hardware* virtualizado de forma simultânea e tinham a impressão de que a máquina física estava disponível para seu uso exclusivo [60].

O ressurgimento da tecnologia de virtualização aconteceu devido a constatação de que a capacidade computacional dos servidores comercializados estava sendo subutilizada. Pesquisas sobre tais tecnologias mostraram que a camada de virtualização inserida entre o *hardware* e a aplicação não compromete o desempenho do sistema como um todo [2].

Nesse contexto, a virtualização pode ser entendida como uma tecnologia que abstrai o *hardware*, permitindo a execução de diversas instâncias de um sistema operacional dentro da mesma máquina física [2]. A Figura 2 [52] apresenta um exemplo de ambiente virtual. Pode-se perceber por meio desta figura a existência de diversos sistemas operacionais sobre um mesmo *hardware*.

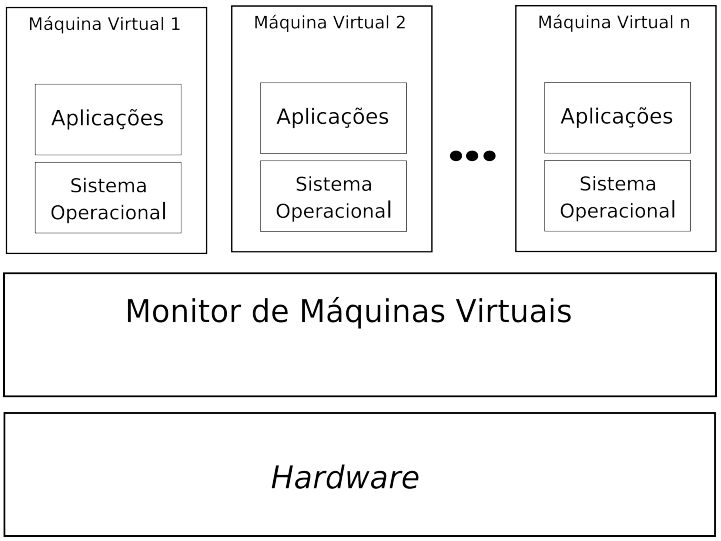

Figura 2: Exemplo de um ambiente que utiliza a virtualização.

Uma máquina virtual é uma cópia de uma máquina física [70]. Cabe ressaltar que as diversas máquinas virtuais localizadas sobre uma determinada máquina real possuem isolamento.

Deste modo, os processos executados em uma máquina virtual específica são independentes das demais. Esta característica permite a maximização da segurança do sistema computacional [2].

O controle e a criação das máquinas virtuais são efetuados por um Monitor de Máquinas Virtuais (MMV). O MMV tem por objetivo disponibilizar uma interface específica para cada máquina virtual, provendo-lhes um ambiente igual ao fornecido por uma máquina física [2, 70]. Além disso, o MMV é responsável por criar máquinas virtuais, as quais se comportam como uma máquina física completa. Tais máquinas virtuais executam seus próprios sistemas operacionais [70].

A tecnologia de virtualização possuem diversas aplicações, descritas a seguir [52]:

- Consolidação de servidores: permite diminuir o número de servidores utilizados para hospedar aplicações, pois a virtualização aumenta a utilização das máquinas físicas.

- Segurança e isolamento: máquinas virtuais são úteis para prover segurança e isolamento. Desta forma, aplicações com um baixo grau de confiança podem ser executadas sem interferir na execução de outras máquinas virtuais. A virtualização pode, deste modo, ajudar a construir ambientes computacionais seguros. A virtualização pode também ser utilizada para o teste de aplicações. Nesse caso, a instabilidade de uma aplicação, hospedada em uma máquina virtual, não afeta as demais máquinas virtuais situadas na mesma máquina física.

- Múltiplos sistemas operacionais: a virtualização provê facilidades para a execução de diversos sistemas operacionais simultaneamente em uma única máquina física. Cada um destes sistemas operacionais pode hospedar diferentes tipos de aplicações.

- Migração de máquinas virtuais: permite a migração de máquinas virtuais de uma máquina física para outra. Tal característica facilita, por exemplo, o balanceamento de carga em um *data center*. Nesse sentido, seria possível realizar a migração de máquinas virtuais hospedadas em servidores sobrecarregados para outro servidor, que possua recursos computacionais disponíveis para receber tal máquina virtual.

Para a execução destes ambientes virtuais, o MMV pode ser inserido em diferentes níveis de uma plataforma computacional. Cada um destes níveis é ilustrado na próxima seção.

2.2.1 Modelos de Virtualização

Esta seção apresenta os principais modelos de virtualização existentes, que se distinguem pelos diferentes níveis onde o virtualizador é inserido no ambiente computacional. Desta forma, esta seção apresenta primeiramente o modelo onde o virtualizador encontra-se no nível de arquitetura do conjunto de instruções. A seguir, a seção aborda o modelo onde o virtualizador

está situado no nível da camada de abstração de *hardware*. Após, a seção ilustra o modelo onde o virtualizador se encontra no nível do sistema operacional. Por fim, esta seção aborda a virtualização no nível de linguagens de programação.

O modelo de virtualização no nível de arquitetura do conjunto de instruções é implementado pela emulação dos conjuntos de instruções de uma arquitetura. Esta emulação é realizada totalmente por *software*. Um computador é formado por processadores, barramentos, memórias, discos rígidos, dispositivos de entrada e saída, entre outros. O conjunto de instruções é a interface entre tais componentes e o sistema operacional.

Desta forma, o emulador recebe as instruções emitidas pelo sistema operacional da máquina virtual e traduz tais instruções com o objetivo de executá-las no próprio *hardware*. Para um emulador realizar sua tarefa com êxito (emular um computador real), este deve ter a capacidade de emular todas as instruções enviadas pela máquina virtual, como a leitura de chips *Read Only Memory* e a reinicialização da máquina.

Tal modelo apresenta vantagens e desvantagens. Pelo lado positivo, o modelo provê facilidades de implementação em múltiplas plataformas. Como o emulador deve traduzir todas as instruções da máquina virtual para a máquina física, a virtualização se torna possível sempre que exista uma forma de traduzir as instruções da máquina virtual para uma instrução correspondente da máquina física. Dessa forma, este modelo pode facilmente hospedar uma máquina virtual com uma arquitetura x86 [68], por exemplo, para plataformas, tais como Alpha [63], Sparc [68], etc. No entanto, a tradução de todas as instruções de uma arquitetura compromete o desempenho do virtualizador que utiliza tal modelo. A Figura 3 [52] apresenta o funcionamento deste modelo. Nesta figura, existe uma máquina virtual, compilada para uma arquitetura x86, operando em um *hardware* com arquitetura Alpha. Como exemplo de virtualizadores que utilizam tal modelo, têm-se o QUEMU [3] e Crusoe [64].

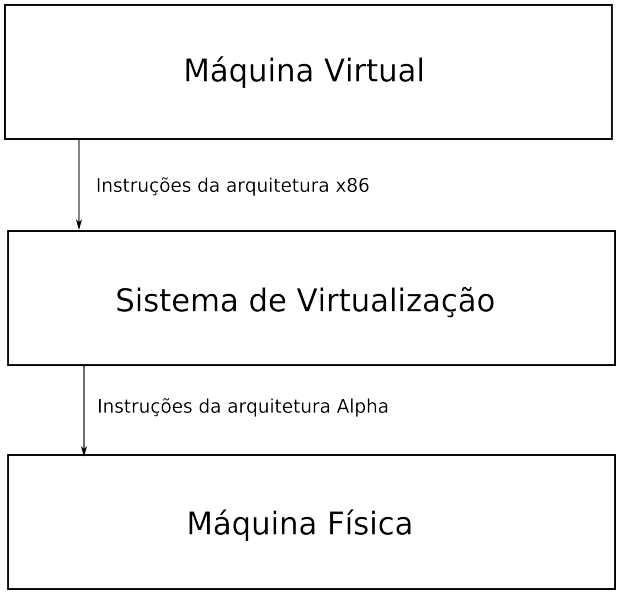

Figura 3: Virtualização no nível do conjunto de instruções.

Já o modelo de virtualização no nível da camada de abstração de *hardware* explora as similaridades entre as arquiteturas das máquinas físicas e virtuais. O objetivo é reduzir o custo de tradução das instruções. Nesse sentido, algumas instruções do sistema operacional das máquinas virtuais são repassadas diretamente para as máquinas físicas. Desta forma, este modelo apresenta um desempenho superior em relação aos virtualizadores que utilizam o modelo de virtualização no nível de arquitetura do conjunto de instruções.

As instruções de arquiteturas, como x86 e compatíveis, normalmente são divididas em duas categorias: instruções privilegiadas e não-privilegiadas. As instruções não-privilegiadas são aquelas que não alteram o estado dos recursos que são compartilhados por vários processos simultâneos, tais como processadores, memória principal e dispositivos de rede [49]. Por outro lado, têm-se as instruções privilegiadas. Tais instruções alteram o estado desses recursos [49].

Um computador pode operar por meio de dois modos: o modo supervisionado e o modo de usuário [49]. No modo supervisionado, qualquer instrução, privilegiada ou não, pode ser executada. Já no modo usuário, apenas as instruções não-privilegiadas podem ser executadas [49]. Dessa forma, o MMV opera sobre o modo supervisionado enquanto que as máquinas virtuais são executadas no modo usuário. Caso uma máquina virtual necessite executar uma instrução privilegiada, uma exceção é gerada [49]. A seguir, o MMV captura tal exceção e emula a instrução.

Assim, as instruções não-privilegiadas são executadas diretamente pelas máquinas virtuais no *hardware*. No entanto, quando uma máquina virtual executa uma instrução privilegiada, é gerada uma exceção, já que as máquinas virtuais operam no modo usuário [49]. O MMV deve então emular a execução da instrução privilegiada. A Figura 4 [52] ilustra este modelo. Pode-se perceber que as instruções privilegiadas são intermediadas pelo virtualizador, enquanto que as não-privilegiadas são acessadas diretamente pelas máquinas virtuais.

O Xen [2] e o VMWare [79] são exemplos de virtualizadores que utilizam tal modelo.

Nos outros modelos apresentados até agora, cada máquina virtual possui uma instância de sistema operacional diferente. No modelo de virtualização no nível do sistema operacional, as máquinas virtuais operam sobre o mesmo sistema operacional da máquina física. Tal modelo cria um ambiente para as máquinas virtuais composto por um sistema operacional (comum a todas as máquinas virtuais), suas aplicações, bibliotecas e um sistema de arquivos. As aplicações destas máquinas virtuais são isoladas em relação aos demais processos do sistema. Além disso, este modelo controla o acesso aos dados de cada máquina virtual. Assim, as máquinas virtuais podem acessar apenas os seus dados. A Figura 5 [52] ilustra este modelo. Pode-se perceber a existência de um único sistema operacional, compartilhado por todas as máquinas virtuais e as aplicações que operam sobre este sistema. Como exemplo de virtualizadores que utilizam tal modelo, têm-se o Jail [62] e User-Mode Linux [24].

O modelo de virtualização no nível de linguagem de programação, o virtualizador é introduzido entre a execução da aplicação e o sistema operacional. Tal modelo provê um conjunto

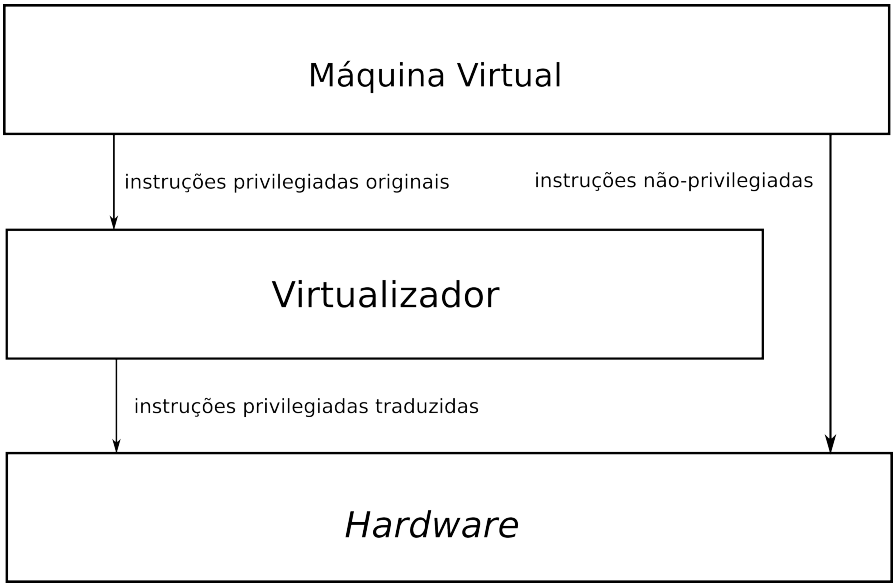

Figura 4: Virtualização no nível da camada de abstração do *hardware*.

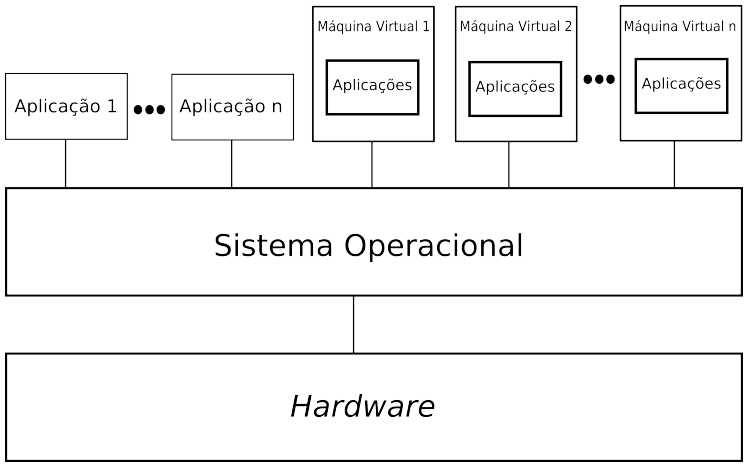

Figura 5: Virtualização no nível do sistema operacional.

de instruções para a aplicação executar. Tais instruções emulam um conjunto de instruções de uma máquina física.

Este modelo permite que uma aplicação seja executada em qualquer sistema operacional, desde que tal sistema tenha suporte ao virtualizador. Além disso, tal modelo garante a independência com relação ao *hardware*. Estas aplicações, executadas neste ambiente, tornam-se extremamente portáveis. No entanto, o desempenho destas aplicações fica prejudicado, já que para cada instrução executada pela aplicação na máquina virtual, existe uma instrução correspondente executada pelo virtualizador. A Máquina Virtual Java [23] é um exemplo de virtualizador que utiliza este modelo.

2.2.2 Monitor de Máquinas Virtuais Xen

Na validação da arquitetura, que será apresentada por meio do Capítulo 5, o virtualizador Xen foi utilizado, pois atende aos requisitos impostos para a concepção da arquitetura. Este virtualizador é disponibilizado de forma gratuita, possui código aberto e um bom desempenho, visto que utiliza o conceito de paravirtualização. Cabe salientar que o Xen utiliza o modelo de virtualização no nível da camada de abstração de *hardware*, apresentado na seção 2.2.1.

A tecnologia de virtualização Xen foi criada por pesquisadores da Universidade de Cambridge. A motivação para o desenvolvimento de tal tecnologia estava em prover uma nova arquitetura de virtualização, que permitisse às máquinas virtuais um desempenho semelhante ao apresentado por um ambiente sem a camada de virtualização. Este virtualizador utiliza a técnica de paravirtualização. Esta técnica não abstrai o *hardware* completamente para os sistemas operacionais que operam sobre o virtualizador [15]. Para que os sistemas operacionais possam executar sobre esta nova plataforma, é necessário que seus *kernels* sejam alterados. Este virtualizador tem suporte aos principais sistemas operacionais, como Linux [7], FreeBSD [47] e Windows [34].

O virtualizador Xen possui três componentes principais: a camada de virtualização Xen, um domínio especial, chamado de domínio 0 e as máquinas virtuais que operam sobre o ambiente virtualizado, chamadas de domínio U (do inglês *unpriviledged*) [2].

A camada de virtualização Xen é inserida sobre o *hardware*. Tal camada é responsável por abstrair os recursos computacionais, como a CPU e a memória. Durante a inicialização do virtualizador, o domínio 0, responsável por hospedar aplicações de gerência que controlam as máquinas virtuais (domínios U), é criado. Este domínio especial é capaz de criar, destruir, pausar e reiniciar uma máquina virtual. Durante a criação de uma máquina virtual é necessário informar a quantidade de memória inicial. Este valor inicial pode ser alterado em tempo de execução.

O virtualizador possui ainda um escalonador que divide o tempo de CPU para as máquinas virtuais. Durante a inicialização do virtualizador, é possível escolher a política de escalonamento mais adequada. Além disso, o percentual de CPU pode ser alterado em tempo de execução por meio de aplicações situadas no domínio 0. O acesso à rede no Xen é gerenciado por um roteador virtual, que redireciona os dados para as interfaces de rede de cada máquina virtual [2].

2.3 *Clusters* de Máquinas Virtuais

Na Seção 2.1, apresentou-se o conceito de *clusters* computacionais. Já na Seção 2.2, definiu-se o conceito de máquinas virtuais. Um *cluster* de máquinas virtuais pode ser entendido como a combinação destas duas tecnologias [12, 31].

Neste sentido, um *cluster* virtual pode ser definido como um conjunto de máquinas virtuais executando sobre nodos físicos conectados por uma rede de qualquer padrão. [5,76]. A Figura 6 apresenta um exemplo de um *cluster* de máquinas virtuais. Nesta figura, percebe-se a presença de um virtualizador em cada nodo do *cluster*. Cada virtualizador gerencia um subconjunto de máquinas virtuais do *cluster* virtualizado.

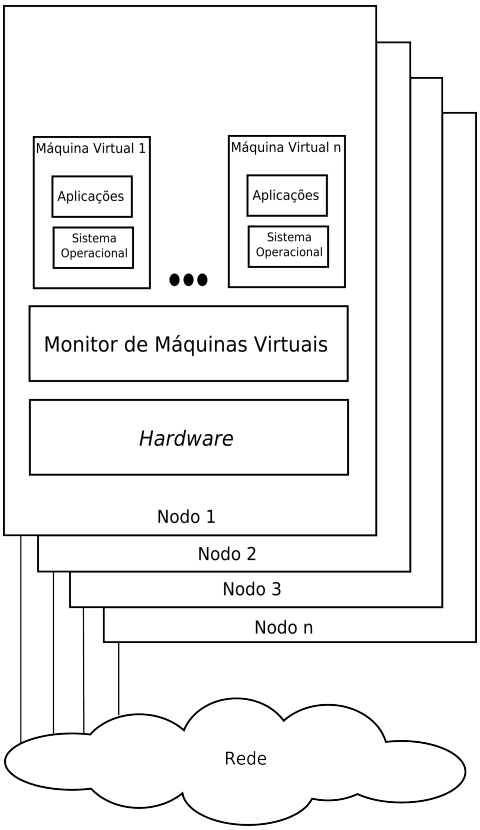

Figura 6: Exemplo de um *cluster* de máquinas virtuais.

Clusters de máquinas virtuais estão sendo utilizados com freqüência, tanto no meio acadêmico quanto no meio corporativo. Exemplos de uso deste tipo de ambiente computacional são apresentados a seguir:

Considere, por exemplo, dois usuários, *A* e *B*. Tais usuários querem utilizar um mesmo recurso *R*, o qual é um *cluster* computacional composto por *n* nodos. Considere ainda que *A* e *B* necessitam de dois ambientes de execução incompatíveis, *Y* e *Z*, respectivamente. Uma possível solução para este problema seria particionar o recurso *R*. Desta forma, o ambiente de execução *Y* do usuário *A* seria instalado sobre *n/2* nodos de *R* e o ambiente *Z* utilizaria a outra metade de *R*. A Figura 7 ilustra tal particionamento.

No entanto, esta abordagem possui alguns problemas. Considere, por exemplo, que em um determinado instante o usuário *A* necessite de todos os recursos de *R*. Considere também que o usuário *B* não esteja utilizando seus recursos. Apesar do usuário *B* não estar usando os recursos

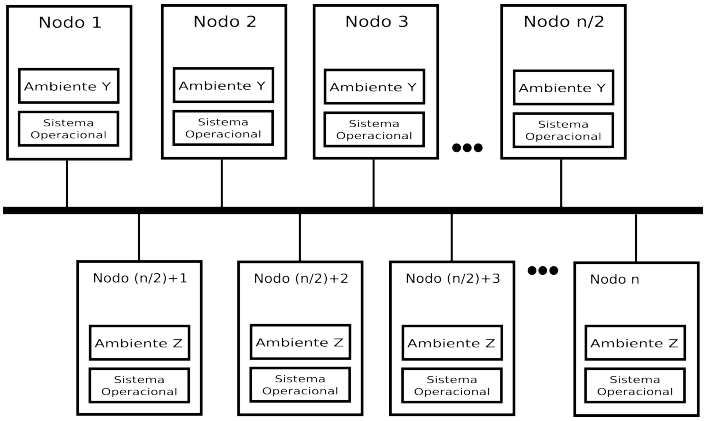

Figura 7: Particionamento do recurso *R*.

de *R*, tais recursos não poderão ser destinados ao usuário *A*, já que os ambientes de execução *Y* e *Z* são incompatíveis e não podem estar instalados em um mesmo nodo.

O uso de um *cluster* de máquinas virtuais permite que os usuários utilizem o recurso *R* de forma eficiente. Desta forma, quando o usuário *B*, por exemplo, não estiver utilizando seus recursos, o usuário *A* poderá utilizá-los, visto que suas aplicações executarão em um ambiente virtual isolado, que possibilita a instalação de bibliotecas e *softwares* sem interferir no ambiente do usuário *B*.

Clusters virtuais também são usados na emulação de aplicações distribuídas, como em [1, 11, 14, 42, 59]. A virtualização permite abstrair o *hardware*, como a memória e processador e aspectos de entrada e saída de um computador, como o armazenamento e a rede.

A virtualização facilita o controle da quantidade de memória, CPU e armazenamento a serem distribuídos para uma determinada aplicação. Além disso, a virtualização facilita, por exemplo, a alteração da quantidade de memória destinada a uma aplicação. O uso de virtualização permite ainda a criação de topologias de redes virtuais entre as máquinas virtuais, tornando possível o uso de um *cluster* computacional conectado por uma rede local como plataforma para testes de aplicações distribuídas.

Desse modo, tem-se o seguinte cenário: um conjunto de máquinas físicas, pertencendo a um *cluster* hospedam, cada uma, um virtualizador. Tal virtualizador gerencia um conjunto de máquinas virtuais. Cada uma destas máquinas virtuais atuará como um nodo do sistema distribuído emulado.

Após a criação das máquinas virtuais sobre o *cluster*, é necessário realizar a conexão entre estas. Assim, é possível implantar o sistema emulado a partir das características do sistema distribuído requerido. Para isso, é preciso configurar a latência e a vazão das conexões entre as máquinas virtuais. A simulação das redes de longa distância é obtida por meio do uso de ferramentas de configuração de rede, como iptables [40].

Clusters de máquinas virtuais são ambientes computacionais complexos. Estes ambientes podem ser formados por dezenas ou centenas de máquinas virtuais, dificultando a gerência de

forma manual. Neste sentido, o próximo capítulo aborda aspectos relacionados com a gerência de *clusters* de máquinas virtuais. Além disso, apresenta-se alguns trabalhos que se propõem a gerenciar este tipo de ambiente. No entanto, estes trabalhos apresentam alguns problemas (abordados na Seção 3.4), motivando a elaboração da arquitetura descrita no Capítulo 4.

3 Gerência de Ambientes Computacionais

Este capítulo está organizado da seguinte forma: a Seção 3.1 apresenta uma visão geral sobre a gerência de ambientes computacionais. Na Seção 3.2, o padrão de gerência utilizado para a elaboração da Arquitetura de Gerenciamento em Ambientes Virtuais é apresentado. A Seção 3.3 aborda os problemas que envolvem o gerenciamento de *clusters* de máquinas virtuais. Por fim, na Seção 3.4, algumas abordagens de gerenciamento de *clusters* de máquinas virtuais são apresentadas.

3.1 Visão Geral

Durante a terceira geração de computadores (entre meados dos anos 60 até o final da década de 70) o ambiente computacional era centralizado [43]. O poder de processamento estava concentrado em um computador de grande porte. Neste contexto, os usuários utilizavam tais computadores por meio de terminais, já que o processamento e o armazenamento de dados eram realizados de forma centralizada através do uso destes computadores. Nesta época, as empresas que vendiam tais computadores também forneciam os *softwares*, como os sistemas operacionais e os sistemas de banco de dados, proporcionando uma grande integração entre os produtos. As atividades de controle e monitoração, como a taxa de utilização da memória, uso da CPU e a quantidade de espaço livre em disco eram simplificadas, uma vez que se restringiam ao único computador de grande porte da empresa. Tais tarefas poderiam ser realizadas de forma manual, pois o ambiente computacional da época limitava-se à máquina de grande porte [61].

Na quarta geração de computadores (a partir do final dos anos 70) o tamanho dos componentes foi reduzido. A redução dos custos e o aumento do poder de processamento dos computadores contribuíram para a mudança de paradigmas, onde o processamento centralizado é substituído pelo processamento distribuído [43].

Segundo Harnedy [36], atualmente os ambientes computacionais são formados por uma grande variedade de *softwares* e *hardwares*. Cada um destes recursos computacionais possui fabricantes distintos. Este alto grau de heterogeneidade, causado pela necessidade de integração entre tecnologias novas e legadas, torna o gerenciamento de ambientes computacionais cada vez mais complexo.

Segundo Heilbronner [38], o crescimento do número de computadores, a distribuição do armazenamento e do processamento das informações e a necessidade cada vez maior de au-

mentar a segurança de acesso a dados, levam a um contexto de extrema heterogeneidade. O crescimento deste ambiente computacional aumenta a probabilidade de ocorrência de falhas e dificulta a identificação destes problemas. A paralisação destes ambientes complexos, mesmo que por alguns instantes, afeta a produtividade das organizações.

Na medida em que o ambiente computacional se torna mais complexo, cresce também a necessidade de um maior controle sobre tal ambiente. Com a mudança de paradigmas, as tarefas de gerência passaram a envolver a administração de diversos recursos computacionais que compõem o vasto ambiente computacional das organizações atualmente.

Para facilitar o gerenciamento destes ambientes computacionais, deve-se utilizar mecanismos para a monitoração e controle. Os padrões de gerenciamento foram desenvolvidos exatamente com este propósito, ou seja, simplificar o processo de administração dos recursos computacionais. Com o gerenciamento desses recursos, através de padrões de gerência, pode-se facilmente administrar a complexidade do ambiente, otimizando a utilização dos componentes disponíveis e diminuindo o tempo de indisponibilidade dos mesmos.

Neste contexto, foram desenvolvidos diversos padrões de gerência, como SNMP, WBEM e TMN (*Telecommunications Management Network*) [32]. O padrão WBEM, escolhido para a elaboração da arquitetura apresentada no Capítulo 4, é abordado na próxima seção.

Esta especificação foi escolhida, pois apresenta uma série de benefícios em relação aos demais padrões de gerência. Existem aspectos complexos de serem gerenciados com outros padrões, como o SNMP. A especificação WBEM possui mecanismos que permitem o controle de aplicações e sistemas [39]. Já o SNMP é mais utilizado para a monitoração de dispositivos de rede. As principais operações deste protocolo (*get* e *set*) são usadas para buscar e alterar informações [76]. Neste protocolo, não existem operações para a invocação de métodos. No padrão WBEM, provedores de métodos são usados com propósito de invocar comandos de aplicações. Neste sentido, os provedores de métodos são usados para invocar operações, como a criação de uma máquina virtual e a inicialização de uma aplicação.

Outra vantagem da especificação WBEM está relacionada com o modelo de dados utilizado. O modelo de dados CIM facilita a representação dos recursos computacionais, uma vez que são usados os conceitos da orientação a objetos. No padrão SNMP, o modelo de informações utilizado é a MIB (*Management Information Base*). A estrutura deste modelo baseia-se nas regras definidas através da SMI (*Structure of Management Information*). Tais regras foram concebidas em uma época onde a maioria das linguagens de programação usadas eram procedurais. Atualmente, linguagens como C++ e Java utilizam os conceitos da orientação a objetos, facilitando a integração entre as aplicações desenvolvidas por meio destas linguagens e o padrão WBEM [39].

Além disso, com a utilização do modelo de informações CIM, é possível estender classes previamente modeladas pela DMTF, simplificando a tarefa de representação dos recursos computacionais a serem gerenciados.

3.2 Especificação WBEM

WBEM é uma arquitetura de gerenciamento elaborada pela DTMF. Esta especificação padroniza a informação (define como o dado é representado) sobre os recursos gerenciados e define o protocolo para monitoração e controle em ambientes distribuídos [45]. Para modelar e representar a informação, esta abordagem utiliza o CIM (*Common Information Model*) [39]. Para a realização do transporte dos dados, primeiramente usa-se o Codificador CIM-XML com o intuito de traduzir os dados CIM em informações XML [27]. Após esta conversão, os dados são efetivamente transportados por meio do protocolo HTTP [39, 45].

Desta forma, esta seção apresenta os principais conceitos relacionados com esta tecnologia. A Subseção 3.2.1 ilustra o funcionamento da arquitetura WBEM. Na Subseção 3.2.2, o modelo de informações CIM é apresentado. A seguir, na Subseção 3.2.3, o protocolo de comunicação WBEM é explicado. A Subseção 3.2.4 apresenta as principais implementações WBEM existentes. Por fim, na Subseção 3.2.5, são apresentados alguns trabalhos que utilizam a especificação WBEM para o gerenciamento de recursos computacionais.

3.2.1 Arquitetura WBEM

A arquitetura WBEM é composta por clientes WBEM, clientes de eventos WBEM, o servidor WBEM e provedores WBEM.

Clientes WBEM realizam requisições e recebem informações sobre os recursos gerenciados [45]. Além disso, os clientes WBEM podem invocar operações de aplicações que estão sendo gerenciadas, como um comando de um virtualizador para criar ou destruir uma máquina virtual.

Os clientes de eventos WBEM recebem informações através de alarmes, os quais têm o objetivo de informar sobre a alteração do estado dos dispositivos gerenciados [45]. Por exemplo, considere que um cliente de eventos deseja receber um aviso quando 80% da memória de uma determinada máquina virtual for utilizada. Quando esta informação extrapolar os limites estabelecidos, será gerado um alarme para o cliente de eventos, com o objetivo de comunicar a alteração significativa no estado do recurso computacional monitorado.

O servidor WBEM tem a função de receber as requisições dos clientes WBEM e interagir com os provedores. O servidor WBEM pode solicitar uma informação sobre algum recurso computacional aos provedores ou solicitar a invocação de alguma operação [39].

Os provedores WBEM interagem diretamente com os recursos gerenciados. Para exemplificar a atuação dos provedores, considere que se necessita saber o estado do *cooler* de um computador. O provedor terá a função de traduzir uma requisição abstrata (retornar o estado do *cooler*) por um comando específico (ler os dois primeiros *bits* do registrador 17, comparar o resultado com 00 (*cooler* com problemas), 01 (*cooler* operando) e 10 (*cooler* funcionando,

mas desligado) e converter o resultado de acordo com o modelo de informações CIM) [39]. É importante salientar que existem 3 tipos de provedores. Os provedores de propriedade servem para coletar informações dos recursos gerenciados. Provedores de métodos são utilizados para invocar uma determinada operação. Por fim, os provedores de indicação são usados para gerar os alarmes. A arquitetura WBEM é ilustrada por meio da Figura 8 [39].

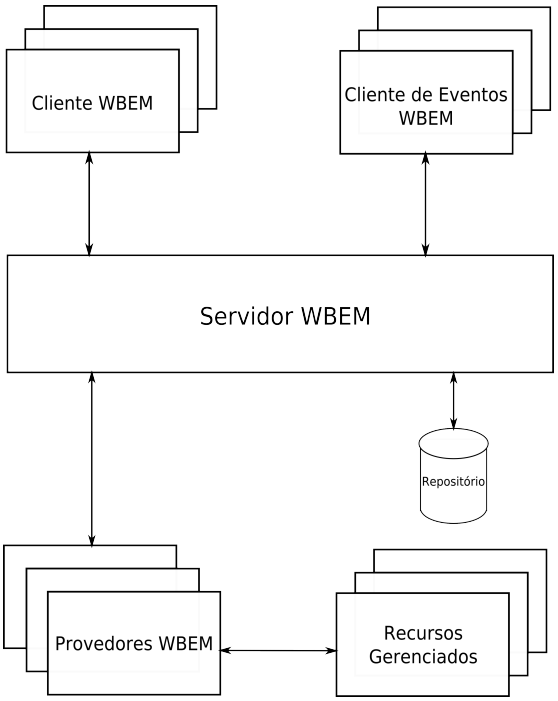

Figura 8: Arquitetura WBEM.

O padrão WBEM foi concebido de modo a prover isolamento entre alguns de seus componentes [39]. Desta forma, os clientes WBEM não necessitam saber como as requisições são conduzidas, ou seja, eles não têm conhecimento sobre a existência dos provedores. De forma análoga, os provedores não precisam saber a origem das operações para realizar o seu trabalho. Este isolamento é garantido, pois os clientes e os provedores WBEM interagem apenas com o servidor WBEM [39, 45]. Assim, caso um cliente WBEM necessite saber a quantidade de espaço livre em disco de um servidor, a requisição (retornar a quantidade de espaço livre em disco) será a mesma, independentemente do sistema operacional que este servidor utiliza. Desta forma, a Figura 9 ilustra este exemplo: o cliente WBEM que requisita a informação comunica-se apenas com o servidor WBEM. De forma análoga, os provedores enviam respostas somente às requisições do servidor WBEM. No entanto, existe a necessidade do desenvolvimento de um provedor WBEM distinto para cada um dos três computadores. Isto porque cada computador possui um sistema operacional diferente. Neste sentido, para o provedor WBEM obter esta informação no sistema operacional Linux, ele deve executar um certo comando. Já para obter esta informação no Windows, o provedor WBEM deve invocar outra operação, própria deste sistema operacional. Ao contrário dos provedores, o cliente WBEM, que requisita a quantidade

de espaço livre em disco do servidor que hospeda o sistema operacional Linux, é o mesmo que requisita a informação para os outros servidores.

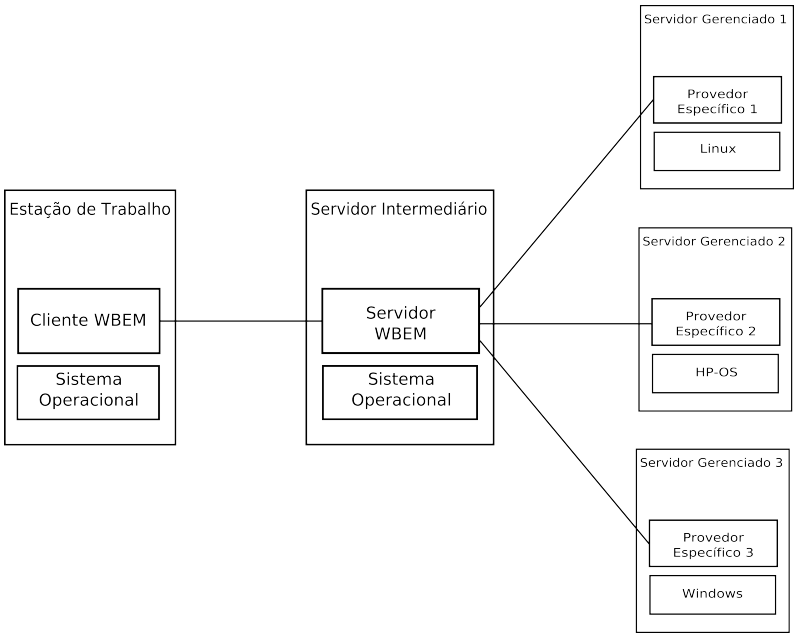

Figura 9: Isolamento entre clientes e provedores WBEM.

O servidor WBEM é o principal componente da especificação WBEM. A Figura 10 [39] ilustra a estrutura deste componente. O servidor WBEM atua como um intermediador, recebendo as requisições dos clientes WBEM e repassando-as aos provedores [33, 39]. O servidor WBEM é formado por uma interface HTTP, utilizada para enviar e receber dados por meio do protocolo HTTP, um codificador e decodificador CIM-XML, responsável pela transformação de uma mensagem XML (*Extensible Markup Language*) recebida via protocolo HTTP em um objeto CIM, por um repositório de informações e pelo gerenciador de objetos CIM (CIMOM).

Quando um cliente WBEM necessita de uma informação, este cria uma requisição e envia ao servidor WBEM por meio do protocolo HTTP. No servidor WBEM, primeiramente, usa-se o decodificador CIM-XML para transformar a mensagem XML em um objeto CIM. Após, o CIMOM consulta o repositório de dados, com o objetivo de encontrar a localização do provedor associado com a informação requisitada. Em seguida, o CIMOM invoca o provedor, e este busca a informação e instancia um objeto CIM. Após, o CIMOM codifica este objeto CIM em um arquivo XML e o envia para o cliente que requisitou as informações. Por fim, o cliente WBEM decodifica este arquivo XML em um objeto CIM e consulta as informações requisitadas [39,45].

É importante salientar que as informações requisitadas por um cliente WBEM podem ser de dois tipos: estáticas e dinâmicas. As informações dinâmicas são obtidas por meio dos provedores WBEM, que interagem com os dispositivos gerenciados. Sabe-se que a localização dos provedores é resgatada por meio de uma consulta ao repositório de informações. No entanto, tal repositório tem outra função. Ele armazena as informações estáticas gerenciadas. Deste modo,

o repositório possui duas atribuições: a primeira é informar ao CIMOM sobre a localização dos provedores que obtêm as informações dinâmicas dos recursos monitorados, a segunda é armazenar informações estáticas dos recursos. Desta forma, quando o CIMOM deseja obter informações estáticas, ele as obtêm por meio do repositório [33, 39].

Para um melhor entendimento com relação à diferença entre informações estáticas e dinâmicas, considere um administrador necessitando saber a quantidade de memória livre de um servidor em um determinado instante. Neste caso, como esta informação pode variar significativamente a cada instante, o CIMOM deve requisitar este dado para o provedor associado. No entanto, caso o administrador necessite saber, por exemplo, a quantidade de processadores de um servidor, o CIMOM buscará esta informação por meio do repositório de dados, pois esta informação não se modifica.

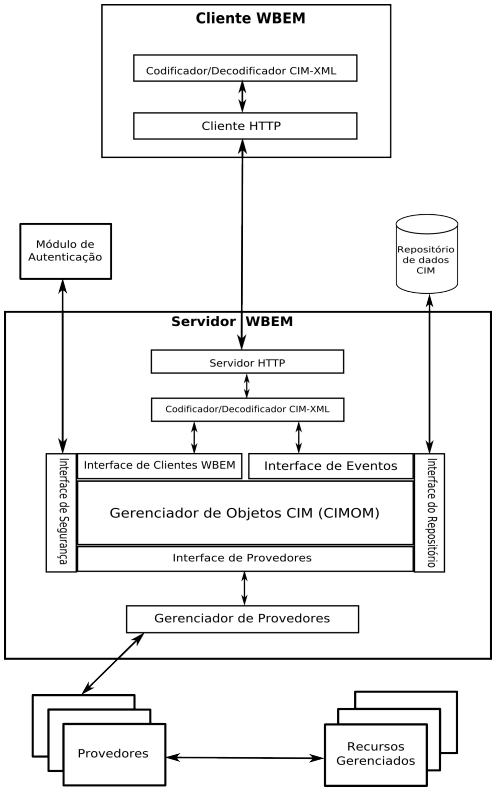

Figura 10: Estrutura do Servidor WBEM.

3.2.2 Modelo de Informações Comum

CIM (Modelo de Informações Comum) é um modelo de informações conceituais para descrever os dados de gerenciamento de um recurso [81]. Este modelo provê uma consistente definição e estrutura por meio do uso de técnicas de orientação a objetos. Em outras palavras, o CIM é uma definição formal utilizada para a representação dos recursos gerenciados. O mo-

delo CIM utiliza diversos conceitos da orientação a objetos, tais como classes, propriedades (atributos), métodos para a manipulação destas propriedades, associações, relacionamentos e herança.

O CIM é composto pela *Specification* CIM e CIM *Schema*. A *Specification* CIM define as regras para a representação dos recursos gerenciados. O CIM *Schema* contém uma série de classes já modeladas que representam os recursos computacionais a serem gerenciados [81].

A *Specification* CIM define um modelo orientado a objetos, baseado na UML (*Unified Modeling Language*) para a representação dos recursos gerenciados [81]. Neste sentido, diversos conceitos da orientação a objetos, como classes, propriedades, métodos e associações são utilizados durante a modelagem de um recurso gerenciado.

A *Specification* CIM define as regras que podem ser utilizadas para representar os recursos gerenciados por meio do *Meta Schema*. A Figura 11 [81] apresenta o *Meta Schema* definido pela DMTF. Nesta figura, é possível observar que o elemento *Class* é a parte principal do modelo. A partir deste elemento, se define um nome para esta classe, um conjunto de propriedades, uma coleção de métodos, seus relacionamentos, associações, especializações e etc [81].

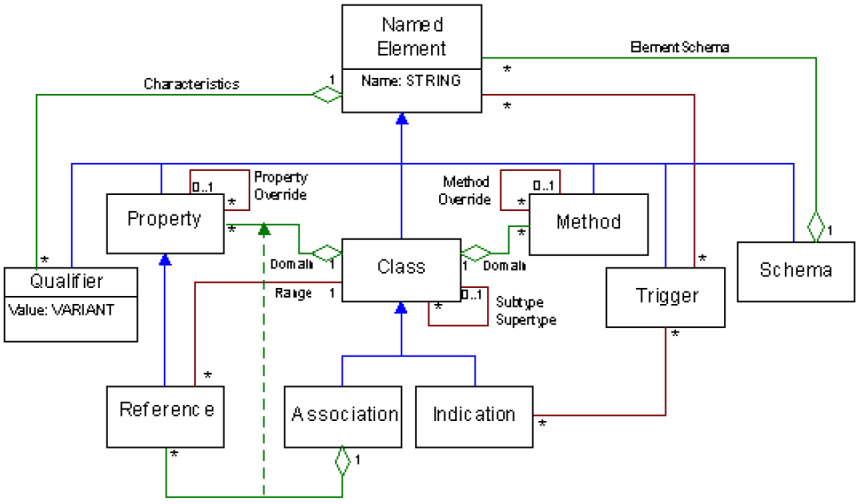

Figura 11: O *Meta Schema* definido pela DMTF.

A *Specification* CIM define também uma linguagem formal para descrever as classes que representam os recursos gerenciados, a MOF (*Managed Object Format*). Desta forma, as classes que representam os recursos são descritas através da linguagem MOF. A Figura 12 apresenta um exemplo de arquivo MOF. Neste arquivo, define-se o nome da classe (*SistemaOperacional*), seus atributos (*identificador*, *nome*, *numeroUsuarios*, *numeroProcessos*, *memoriaLivre*) e os seus métodos (*reiniciarSistema*, *desligarSistema*).

O CIM *Schema* fornece um conjunto de classes e associações. Tais classes são estendidas com objetivo de gerenciar ambientes específicos. Por exemplo, o CIM *Schema* contém uma classe chamada *CIM_OperatingSystem*. Tal classe contém atributos comuns a todos sistemas

```
class SistemaOperacional
{
    // atributos
    [Key] uint32 identificador;
    string nome;
    uint32 numeroUsuarios;
    uint32 numeroProcessos;
    uint32 memoriaLivre;

    //métodos
    boolean reiniciarSistema( );
    boolean desligarSistema( );

};
```

Figura 12: Arquivo que modela as informações de interesse de um sistema operacional.

operacionais. Considere agora que se necessite gerenciar atributos e métodos de um sistema operacional que não foram especificados na classe *CIM_OperatingSystem*. Neste sentido, é preciso desenvolver uma classe, que chamaremos de *MeuSistemaOperacional*. Esta nova classe deve estender a classe *CIM_OperatingSystem*. Além disso, todas as informações que se deseja gerenciar devem ser representadas nesta nova classe. A Figura 13 ilustra este exemplo. Nesta figura, pode ser observado um arquivo MOF que cria esta nova classe. Além disso, pode-se perceber os atributos e métodos específicos desta classe que não existem na classe *CIM_OperatingSystem*.

```
class MeuSistemaOperacional : CIM_OperatingSystem
{
    // atributos específicos deste SO
    uint32 espacoLivreEmDisco;
    uint32 espacoUsadoEmDisco;
    uint32 espacoTotalDoDisco;

    //métodos
    boolean instalarPacote(string nomePacote);

};
```

Figura 13: Exemplo de um arquivo MOF que estende a classe *CIM_OperatingSystem.*

O CIM *Schema* é composto pelo *Core Model*, pelo *Common Model* e pelo *Extension Schema* [81]. O *Core Model* captura noções que são aplicáveis a todas áreas de gerenciamento. O *Core Model* é uma coleção de classes, associações, atributos e métodos que fornece um vocabulário básico para a representação do universo a ser gerenciado [81].

O *Common Model* estende o *Core Model* para representar noções específicas de determinadas áreas, mas independente de uma tecnologia ou implementação particular [81].

Já o *Extension Schema* estende *Common Model* para gerenciar ambientes específicos dependentes de tecnologia.

Considerando ainda o exemplo da Figura 13, a classe *CIM_OperatingSystem* encontra-se dentro do *Common Model*, enquanto que a classe criada (*MeuSistemaOperacional*), encontra-se dentro do *Extension Schema*.

3.2.3 Protocolo de Comunicação WBEM

Conforme a definição da especificação WBEM [29], a comunicação entre os seus componentes pode ser realizada por meio de dois protocolos: CIM-XML [27] e WS-Management [30]. No entanto, o protocolo mais comumente utilizado é o CIM-XML. Por este motivo, somente este protocolo será abordado nesta seção.

Para um melhor entendimento com relação ao processo de comunicação da especificação WBEM, considere que se necessita obter a quantidade de processos executando sobre o sistema operacional de um servidor. Na Figura 14, pode-se observar os componentes responsáveis pelo processo de comunicação da especificação WBEM. Primeiramente, a requisição realizada pelo cliente WBEM é traduzida em um arquivo XML pelo Codificador CIM-XML. Um exemplo de arquivo XML pode ser visto através da Figura 15. Este arquivo tem o objetivo de requisitar o número de processos do sistema operacional. Após, esta requisição é enviada ao servidor WBEM por meio do protocolo HTTP. A seguir, o servidor WBEM analisa a requisição recebida e busca a informação. Após, a resposta para o cliente WBEM é enviada. Para isto, usa-se o Codificador CIM-XML, que transforma o objeto CIM em um arquivo XML. O arquivo XML que contém a informação requerida pelo cliente WBEM pode ser observado por meio da Figura 16. Conforme pode ser observado através desta figura, o valor retornado foi 63, significando que existiam 63 processos sendo executados naquele instante.

3.2.4 Implementações WBEM

Nesta seção, serão abordadas as implementações OpenPegasus [54], WMI (*Windows Management Instrumentation*) [75], WBEM Services [80] e Open WBEM [55]. Estas implementações possuem os mesmos componentes descritos na arquitetura WBEM. Por isso, esta seção não apresentará os detalhes de cada abordagem, já que o funcionamento de tais implementações baseia-se na especificação abordada durante este capítulo.

A Tabela 1, adaptada do trabalho de Lee [45], ilustra as características de cada implementação WBEM. Os aspectos comparados são: portabilidade, linguagem de programação usada para

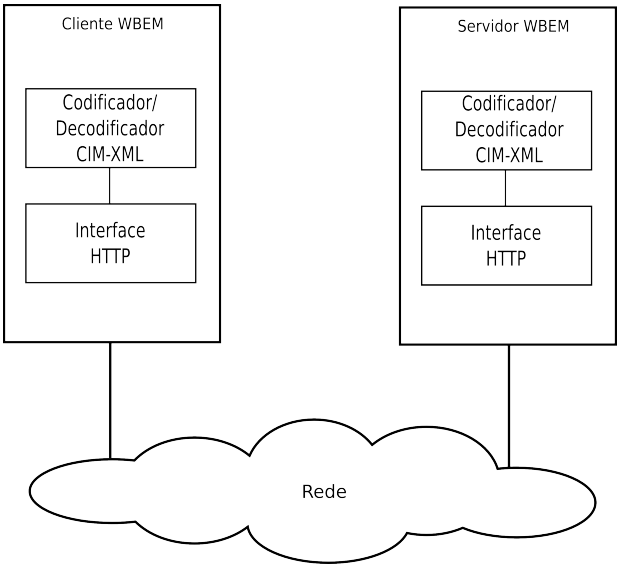

Figura 14: Processo de comunicação da especificação WBEM

```
<?xml version="1.0" encoding="utf-8" ?>
<CIM CIMVERSION="2.0" DTDVERSION="2.0">
 <MESSAGE ID="82000" PROTOCOLVERSION="1.0">
  <SIMPLEREQ>
   <IMETHODCALL NAME="GetProperty">          <- Operação de interesse
    <LOCALNAMESPACEPATH>
     <NAMESPACE NAME="root" />               <- Namespace de interesse
     <NAMESPACE NAME="cimv2" />
    </LOCALNAMESPACEPATH>
    <IPARAMVALUE NAME="InstanceName">
     <INSTANCENAME CLASSNAME="CIM_OperatingSystem">   <- Classe de interesse
      <KEYBINDING NAME="CSCreationClassName">
       <KEYVALUE VALUETYPE="string">CIM_UnitaryComputerSystem</KEYVALUE>
      </KEYBINDING>
      <KEYBINDING NAME="CSName">
       <KEYVALUE VALUETYPE="string">system1.hp.com</KEYVALUE>   <- Identificador de interesse
      </KEYBINDING>
      <KEYBINDING NAME="CreationClassName">
       <KEYVALUE VALUETYPE="string">CIM_OperatingSystem</KEYVALUE>
      </KEYBINDING>
      <KEYBINDING NAME="Name">
       <KEYVALUE VALUETYPE="string">HP-UX</KEYVALUE>
      </KEYBINDING>
     </INSTANCENAME>
    </IPARAMVALUE>
    <IPARAMVALUE NAME="PropertyName">        <- Propriedade de interesse
     <VALUE>NumberOfProcesses</VALUE>
    </IPARAMVALUE>
        ●●●
   </IMETHODCALL>
  </SIMPLEREQ>
 </MESSAGE>
</CIM>
```

Figura 15: Arquivo XML que requisita o número de processos ao servidor WBEM.

a implementação de clientes WBEM e provedores, licença, sistemas operacionais suportados, organização que a desenvolveu, documentação disponibilizada para o uso da implementação, e se possui código aberto.

De acordo com a análise realizada por meio desta tabela, foi possível observar os pontos fortes e fracos de cada implementação. OpenPegasus e WBEM Services são implementações com licença de código aberto. Além disso, ambas foram desenvolvidas por organizações que participam ativamente da padronização e do aperfeiçoamento da especificação WBEM. Segundo

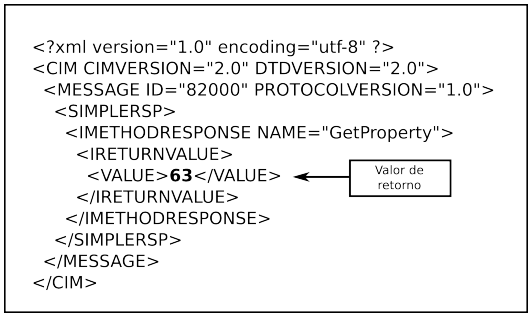

Figura 16: Resposta enviada para o cliente WBEM.

Tabela 1: Comparação entre as principais implementações WBEM.

	OpenPegasus	WBEM Sevices	WMI	Open WBEM
Portabilidade	Boa	Excelente	Ruim	Boa
Linguagem	C++	Java	Plataforma .NET	C++
Licença	MIT Open	SISSL v1.2	Microsoft	BSD License
Plataformas	Linux, Windows	Qualquer	Windows	Linux, Windows
Desenvolvedor	The Open Group	Sun Microsystems	Microsoft	Caldera INC
Documentação	Excelente	Excelente	Excelente	Ruim
Código livre	Sim	Sim	Não	Sim

testes de Lee [45], WBEM Services e OpenPegasus são as implementações com menor latência entre a requisição de um cliente até o recebimento da resposta. No entanto, entre estas duas abordagens, OpenPegasus apresentou um desempenho superior se comparado ao desempenho da implementação WBEM Services. Com relação à linguagem de programação utilizada para o desenvolvimento dos provedores e clientes WBEM, WBEM Services requer o uso da linguagem Java [23], enquanto que OpenPegasus requer a utilização da linguagem C++ [21].

A implementação WMI, desenvolvida pela Microsoft, possui como pontos positivos uma boa documentação e facilidades no desenvolvimento dos provedores e clientes WBEM. No entanto, esta implementação requer o uso da plataforma Windows (fraca portabilidade) e não pode ser utilizada livremente. Ainda, WMI possui um desempenho inferior se comparado às implementações OpenPegasus e WBEM Services [45].

A implementação Open WBEM restringe o uso da linguagem de programação C++ para o desenvolvimento de provedores e clientes WBEM. Além disso, tal implementação possui um desempenho inferior ao OpenPegasus e WBEM Services [45].

Para validar a arquitetura apresentada no Capítulo 4, a implementação WBEM escolhida foi OpenPegasus, pois apresenta diversas vantagens: é um *software* de código aberto, é utilizado em sistemas de grandes corporações, como a IBM (IBM Director [41]) e HP (HP-UX [58]), e segundo testes realizados por Lee [45], é a implementação WBEM com a menor latência entre a requisição de um cliente até o recebimento da resposta.

3.2.5 Estudos de Casos

Esta seção apresenta trabalhos que utilizam a especificação WBEM para gerenciar recursos computacionais. Primeiramente, é apresentado o trabalho descrito por Tan [71]. Este trabalho utiliza a especificação WBEM para gerenciar *storages*. Após, o IBM Director [41] é abordado. Esta plataforma é utilizada para gerenciar ambientes computacionais, como máquinas físicas que hospedam diferentes sistemas operacionais e máquinas virtuais.

A dificuldade de gerenciar informações armazenadas em *storages* aumentou significativamente com a queda dos custos para a aquisição destes equipamentos. Esta redução dos preços influenciou o rápido crescimento do volume de dados com necessidade de armazenamento. Além do crescimento do volume das informações, outro fato que dificulta a administração é a heterogeneidade dos *storages*, já que cada um apresenta uma interface de gerenciamento específica. Outro problema existente está relacionado com a escalabilidade destes equipamentos, já que as interfaces proprietárias de *storages* comprometem a integração de tais dispositivos [71].

Diante deste contexto, o trabalho elaborado por Tan [71] apresenta a implementação de provedores WBEM para o *storage* Sun StoEdge 3510, com objetivo de facilitar a integração deste produto com outros *storages* existentes. Para isso, foi realizado o mapeamento da interface do Sun StoEdge 3510 para uma interface comum, a SMI-S (*Storage Management Initiative-Specification*) [65].

A especificação SMI-S, proposta pela SNIA (Storage Networking Industry Association) [66], define uma interface segura, extensível e interoperável, visando facilitar o gerenciamento de sistemas de *storages* heterogêneos. Tal especificação utiliza a arquitetura WBEM como protocolo de comunicação entre as aplicações de gerenciamento e os *storages* gerenciados. Assim, a representação dos *storages* gerenciados é feita por meio do modelo de dados CIM. O padrão SMI-S foi desenvolvido para integrar-se ao conjunto de modelos já especificados no CIM *Schema*. O SMI-S está situado entre as aplicações de gerência e os objetos gerenciados (*storages*). Desta forma, as operações realizadas por uma aplicação para monitorar *storages* distintos será a mesma. Por exemplo, considere que se deseja monitorar a quantidade de espaço livre em um *storage*. Tal operação é realizada por um cliente WBEM (presente na aplicação de gerenciamento) e encaminhada ao servidor WBEM. Nesse momento, o servidor WBEM invoca o provedor específico do *storage* que se deseja a informação. Esta consulta abstrata (retornar a quantidade de espaço livre do *storage*) é traduzida em uma operação específica (invocar o comando da interface de gerenciamento específica do *storage*).

Os provedores WBEM implementados nesse trabalho fornecem um mapeamento entre a interface SMI-S e o dispositivo de *storage*. Tais provedores provêem informações de monitoração às aplicações de gerenciamento, permitem a descoberta de *storages* pelas aplicações e disparam alarmes que informam sobre alguma alteração do estado do *storage* [71].

O segundo estudo de caso aborda a plataforma IBM Director [41]. Cabe salientar que esta ferramenta possui alguns módulos que não serão mencionados nesta seção, pois não utilizam a especificação WBEM como protocolo de comunicação.

O crescimento das organizações exige cada vez mais poder computacional. Na medida que mais recursos computacionais são incluídos, maior será a complexidade e o custo de gerenciamento. Este crescimento exige ainda um incremento proporcional sobre a necessidade de controle das informações [41].

Existem problemas complexos de administrar-se manualmente, exigindo a contratação de operadores experientes a um custo elevado. Ainda assim, existe o risco deste administrador não executar a tarefa de maneira eficaz. Então, torna-se evidente a necessidade de usar um mecanismo automatizado de gerenciamento, pois reduz consideravelmente os custos de uma organização e torna a monitoração dos recursos mais eficiente.

Diante deste contexto, a plataforma IBM Director foi desenvolvida com objetivo de simplificar o gerenciamento de ambientes computacionais heterogêneos. Com o IBM Director, os administradores de TI (Tecnologia da Informação) podem monitorar de forma remota recursos, tais como servidores, estações de trabalho, computadores móveis (*notebooks*), dentre outros. A ferramenta permite também a monitoração de componentes como processadores, discos e memória [41].

O IBM Director é formado por 4 componentes: o IBM Director Server, o IBM Director Console e o IBM Director Core Services e o IBM Director Agent [41]. No entanto, este último não utiliza WBEM como protocolo de comunicação, e portanto, não será abordado nesta seção.

O IBM Director Server, o principal componente do IBM Director, deve ser instalado sobre um servidor de gerenciamento. Este componente provê funções básicas, como a descoberta de objetos gerenciados, armazenamento da configuração e das informações dos recursos gerenciados e o controle de eventos. O IBM Director Server armazena os dados em um banco de dados SQL (*Structured Query Language*). Desta forma, é possível acessar as informações dos objetos gerenciados mesmo quando estes não se encontram disponíveis [41].

O IBM Director Core Services é instalado sobre os objetos gerenciados. Tal componente permite a monitoração de componentes como a CPU, a memória, o disco, a reinicialização do sistema gerenciado e a monitoração de eventos. Este componente comunica-se com o O IBM Director Server por meio do protocolo WBEM [41].

IBM Director Console deve ser instalado sobre a estação de trabalho do operador que gerencia os recursos. Este componente obtém informações sobre os objetos gerenciados comunicando-se com o IBM Director Server. O IBM Director Console provê uma interface gráfica que permite aos usuários controlar de forma integrada todos os recursos gerenciados [41].

3.3 Gerência de *Clusters* de Máquinas Virtuais

A tecnologia de virtualização cria uma nova camada entre o *hardware* e a aplicação de usuário [35]. Este novo nível de abstração aumenta a complexidade no gerenciamento de máquinas virtuais e aplicações que operam sobre este ambiente, uma vez que com o uso de virtualização, passou-se a contar com múltiplas máquinas virtuais executando sobre a mesma plataforma de *hardware*, enquanto que a abordagem tradicional assume que haverá um único sistema operacional sobre a plataforma, tendo acesso exclusivo a todos os seus recursos.

A virtualização, através desta nova camada, cria um nível a mais de indireção [35]. Para controlar e monitorar um *cluster* que não utiliza a tecnologia de virtualização, é necessário apenas a referência para as máquinas físicas do ambiente. Entretanto, para gerenciar um *cluster* de máquinas virtuais necessita-se da referência para a máquina física e a referência para a máquina virtual. A Figura 17 ilustra esta diferença. Cabe salientar que a Figura 17(a) representa um ambiente que não utiliza a tecnologia de virtualização, enquanto que a Figura 17(b) utiliza. Considere ainda que a aplicação de gerência, localizada no *front-end*, deseja reiniciar uma máquina do ambiente. Desta forma, para a aplicação reiniciar uma máquina no ambiente representado pela Figura 17(a) é necessária apenas a referência da máquina em questão. Já no ambiente representado pela Figura 17(b), tal aplicação deve primeiramente localizar o *host* que esta máquina virtual se encontra. Em seguida, a aplicação deve invocar o comando responsável por reiniciar a máquina virtual. Ainda na Figura 17(b), cabe salientar que as setas tracejadas representam que a aplicação necessitou pesquisar a localização da máquina virtual em questão antes de reiniciá-la. Já a seta contínua representa a invocação do comando na máquina virtual desejada.

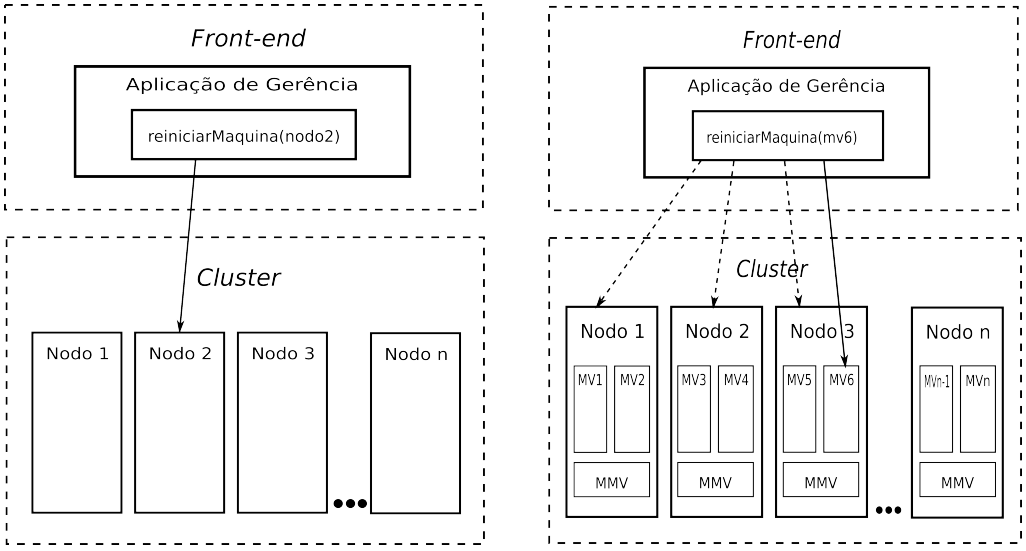

Figura 17: (a) Ambiente sem virtualização. (b) Ambiente que utiliza a virtualização.

Em um *cluster* virtual, cada um dos seus *hosts* possuem diversas máquinas virtuais. Com isso, a aplicação que interage com tal ambiente precisa ter a informação sobre o *host* do *cluster* que está hospedando uma determinada máquina virtual antes de invocá-la. Deste modo, a arquitetura deste trabalho, detalhada no Capítulo 4, será responsável por diminuir esta complexidade imposta pela virtualização. Neste contexto, a aplicação de gerência se relacionará com os serviços disponibilizados pela arquitetura e não com o *cluster* virtualizado. Dessa forma, tal aplicação informará apenas o identificador da máquina virtual. É função da arquitetura verificar em qual *host* do *cluster* a máquina virtual está localizada.

Outro problema encontrado na gerência de um *cluster* de máquinas virtuais está relacionado com a construção e configuração do ambiente virtual. Um ambiente com muitas máquinas virtuais dificulta a configuração manual. Considere, por exemplo, um *cluster* composto por 8 nodos físicos. Suponha que se necessite criar sobre este *cluster* um ambiente com 48 máquinas virtuais. Considere ainda que exista a necessidade de disparar aplicações em cada uma destas máquinas virtuais e tais aplicações exigem a alteração em determinados arquivos de configuração. Na etapa de criação das máquinas virtuais, seria preciso criar um arquivo de configuração para cada uma das máquinas virtuais, informando características como quantidade de memória, nome e o IP das máquinas virtuais. Ainda nesta etapa, seria necessário gerenciar as imagens utilizadas por cada máquina virtual. Por fim, seria necessário invocar o comando do virtualizador responsável por criar cada máquina virtual. Para realizar tal tarefa, o virtualizador utiliza os arquivos de configuração previamente criados. Após a criação do ambiente virtual, seria necessário acessar cada uma destas 48 máquinas virtuais, alterar os arquivos de configuração e disparar as aplicações.

Por meio deste exemplo, pode-se notar que o gerenciamento manual de um *cluster* de máquinas virtuais é complexo.

Algumas abordagens se propõem a resolver os problemas relacionados com o gerenciamento de *clusters* de máquinas virtuais. No entanto, estas abordagens não os resolvem de forma satisfatória. A próxima seção apresenta estes trabalhos. Ainda nesta seção, os problemas destes trabalhos, que motivaram a elaboração da arquitetura apresentada por meio do Capítulo 4, são analisados.

3.4 Outras Abordagens para a Gerência de *Clusters* de Máquinas Virtuais

O trabalho proposto por Krsul [44] provê o gerenciamento de máquinas virtuais para um ambiente de grades computacionais. A arquitetura deste trabalho fornece 3 tipos de serviços: a criação de máquinas virtuais, informações sobre o ambiente virtualizado e a destruição de máquinas virtuais ativas. As informações obtidas a partir do ambiente são utilizadas na etapa

anterior à criação de novas máquinas virtuais. Quando a arquitetura recebe uma requisição, primeiramente é verificado se existem recursos disponíveis, como memória livre e espaço em disco. Caso existam recursos suficientes, a máquina virtual será criada. Clientes (usuários ou aplicações) invocam os serviços disponibilizados por esta arquitetura. A comunicação entre os componentes da arquitetura é realizada por meio de *sockets* [77]. Já a comunicação entre a arquitetura e a aplicação que utiliza os serviços é feita por meio de arquivos XML. Estes arquivos contêm as informações necessárias para a criação das máquinas virtuais. A implementação deste trabalho usou os virtualizadores VMware e User-Mode Linux.

McNett [48] define um *framework* para o gerenciamento de *clusters* de máquinas virtuais. O objetivo deste trabalho está relacionado com a disponibilização de serviços para a gerência de máquinas virtuais de um *cluster*. Estes serviços, que são invocados por clientes (aplicações), permitem a criação, a reinicialização, a destruição, a migração, a pausa e a retomada de máquinas virtuais. Além disso, o *framework* disponibiliza serviços para monitoração das máquinas virtuais, como a utilização de CPU, o uso da memória e o controle sobre o estado das máquinas virtuais. Este trabalho possui também um serviço que notifica as aplicações sobre a mudança de estado de uma determinada máquina virtual. A implementação desse trabalho utilizou o virtualizador Xen.

O trabalho proposto por Bhatia [5] define uma ferramenta para a administração de um *cluster* de máquinas virtuais. Os usuários interagem com tal ferramenta por meio de uma interface gráfica. Este trabalho facilita a gerência de um *cluster* virtual, permitindo a migração de máquinas virtuais de um *host* físico para outro, o balanceamento automático de carga e a destruição de máquinas virtuais. Além disto, a ferramenta mostra informações de monitoração, como a lista de máquinas virtuais em cada *host* do *cluster*, o estado das máquinas virtuais, utilização de CPU e a quantidade de memória disponibilizada para cada máquina virtual do ambiente. Este trabalho foi implementado por meio da linguagem de programação C [22] e a comunicação entre os componentes é realizada através de Chamadas Remotas de Funções (RPC). O virtualizador utilizado no protótipo desta ferramenta foi o Xen.

O trabalho elaborado por Park [56] tem o objetivo de gerenciar um *cluster* de máquinas virtuais. Para tanto, uma interface gráfica foi desenvolvida para interagir com os administradores do ambiente. Os componentes da arquitetura deste trabalho comunicam-se por meio do padrão WBEM. Através desta interface, os usuários podem gerenciar o ciclo de vida das máquinas virtuais (criar, reiniciar, destruir e migrar). Além disto, existe um componente responsável por monitorar os recursos dos *hosts* do *cluster*. Este trabalho usa o componente de monitoração para verificar se o *host* que hospedará uma nova máquina virtual possui recursos suficientes. Desta forma, antes de criá-la, recursos como a memória disponível e a quantidade de espaço livre em disco do *host* são verificados.

Os trabalhos abordados nesta seção apresentam alguns problemas. Nenhum dos trabalhos é completo o suficiente para disponibilizar serviços para a construção automatizada de um ambiente virtual, desde a criação das máquinas virtuais, o repasse automático de arquivos de confi-

guração para o ambiente virtual e a execução das aplicações que executam sobre as máquinas virtuais do *cluster*. Além disso, muitos destes trabalhos fornecem uma interface gráfica para interagir com os administradores do ambiente, inviabilizando o uso destes trabalhos por aplicações de gerência.

Neste sentido, a arquitetura que será apresentada no próximo capítulo tem por objetivo minimizar os problemas apresentados anteriormente. Uma série de serviços, invocados por aplicações de gerência, são disponibilizados pela arquitetura. Estes serviços permitem a monitoração das máquinas virtuais, a gerência do ciclo de vida das máquinas virtuais e o controle sobre as aplicações que são executadas no ambiente virtual. O uso combinado destes serviços permite a construção automática e a reconfiguração de um *cluster* de máquinas virtuais.

4 Arquitetura de Gerenciamento em Ambientes Virtuais

Os trabalhos apresentados na seção anterior não permitem o controle e a monitoração das aplicações que operam sobre o ambiente virtualizado. Além disso, a maioria dos serviços de monitoração não utiliza o conceito de alarmes. Os serviços que disponibilizam mecanismos de notificação liberam as aplicações da necessidade periódica de verificar a taxa de utilização de um determinado recurso ou a mudança de estado de uma máquina virtual. Além disto, os serviços dos trabalhos apresentados na Seção 3.4, usados por clientes (aplicações), possuem alguns problemas. A invocação de um serviço destes trabalhos corresponde a uma operação sobre apenas uma máquina virtual. Ou seja, não existem serviços que, quando invocados, efetuem uma operação sobre um conjunto de máquinas virtuais.

A maior parte dos trabalhos existentes não fornece uma interface de serviços, o que inviabiliza a utilização destes trabalhos para o desenvolvimento de aplicações de gerência. Apesar de existirem alguns trabalhos que fornecem serviços para aplicações de gerência, estes serviços, muitas vezes, não são suficientes para, por exemplo, automatizar a construção de um ambiente distribuído emulado. Neste caso, torna-se necessário que as aplicações implementem mecanismos para automatizar tal ambiente, criando soluções de gerenciamento específicas e propícias a erros.

Na Seção 3.3 observou-se os problemas envolvidos na gerência de *clusters* de máquinas virtuais. Na Seção 3.4 viu-se que os trabalhos existentes não são completos o suficiente para suprir as necessidades apresentadas na Seção 3.3. As limitações destes trabalhos motivaram a elaboração da arquitetura apresentada neste capítulo, que deve atender aos requisitos a seguir:

1. Capacidade de prover serviços para aplicações de gerência, permitindo que estas aplicações se preocupem apenas na resolução de seus problemas. Estes serviços devem facilitar o gerenciamento do *cluster* de máquinas virtuais através da abstração criada pela arquitetura.

2. Os serviços disponibilizados pela arquitetura devem permitir a construção de um *cluster* de máquinas virtuais de forma automática. Desta forma, serviços para a criação de máquinas virtuais, para a invocação de aplicações e para o envio de arquivos de configuração deverão ser criados. Além disso, a arquitetura deve prover serviços que permitam a reconfiguração do ambiente. Neste sentido, serviços para a recriação e destruição das máquinas virtuais e para monitoração do ambiente (tanto por *polling* quanto por meio de alarmes) também deverão ser criados.

3. Os serviços da arquitetura devem simplificar a concepção das aplicações de gerência. Desta forma, a invocação de um serviço da arquitetura deverá efetuar uma operação sobre um conjunto de máquinas virtuais. Como exemplo, pode-se citar o serviço que, quando invocado, destrói todas as máquinas virtuais do *cluster*.

4. A arquitetura deve utilizar um padrão de gerência. Este padrão deve ser capaz minimizar a complexidade imposta pelos ambientes distribuídos, facilitando a concepção da arquitetura.

Esta arquitetura disponibiliza serviços para o controle do ciclo de vida e monitoração de máquinas virtuais. Além disso, esta arquitetura provê serviços para o controle de aplicações que executam sobre as máquinas virtuais do *cluster*. Tais serviços facilitam a gerência deste *cluster* virtual, pois o ambiente distribuído é abstraído. Uma aplicação de gerência, quando utiliza os serviços desta arquitetura, não tem a informação sobre a localização das máquinas virtuais nos diversos nodos do *cluster*. Desta forma, a gerência das máquinas virtuais ocorre como se estas estivessem situadas na mesma máquina física que a aplicação de gerência, tornando transparente o acesso às máquinas virtuais localizadas remotamente.

A comunicação entre os componentes desta arquitetura é realizada por meio da especificação WBEM. Tal padrão diminui a complexidade imposta pelos ambientes computacionais distribuídos, através da definição de um modelo de dados comum e um protocolo para o transporte destas informações.

Este capítulo está organizado da seguinte forma: a Seção 4.1 mostra a disposição física dos componentes que constituem a Arquitetura de Gerenciamento em Ambientes Virtuais. Na Seção 4.2, um detalhamento dos módulos que constituem os Serviços de Gerenciamento em Ambientes Virtuais é apresentado. Por fim, os serviços disponibilizados pela arquitetura são descritos.

4.1 Disposição Física dos Componentes da Arquitetura

A disposição física dos componentes da arquitetura são ilustrados por meio da Figura 18. Esta arquitetura possui dois componentes principais, os Serviços de Gerenciamento em Ambientes Virtuais, localizado no *front-end* do *cluster*, e o Gerenciador de Ambientes Virtuais, situado em um ponto a partir do qual ele possa executar operações de gerência sobre as máquinas virtuais. Para fins de compreensão, este ponto será chamado de nível privilegiado. Além disso, pode-se notar a presença de um componente denominado Aplicação de Gerência, que utiliza os serviços disponibilizados pela arquitetura. A seguir, cada um destes componentes é apresentado.

O componente chamado Aplicação de Gerência invoca serviços da arquitetura por meio de uma interface com o componente de Serviços de Gerenciamento em Ambientes Virtuais. Tal

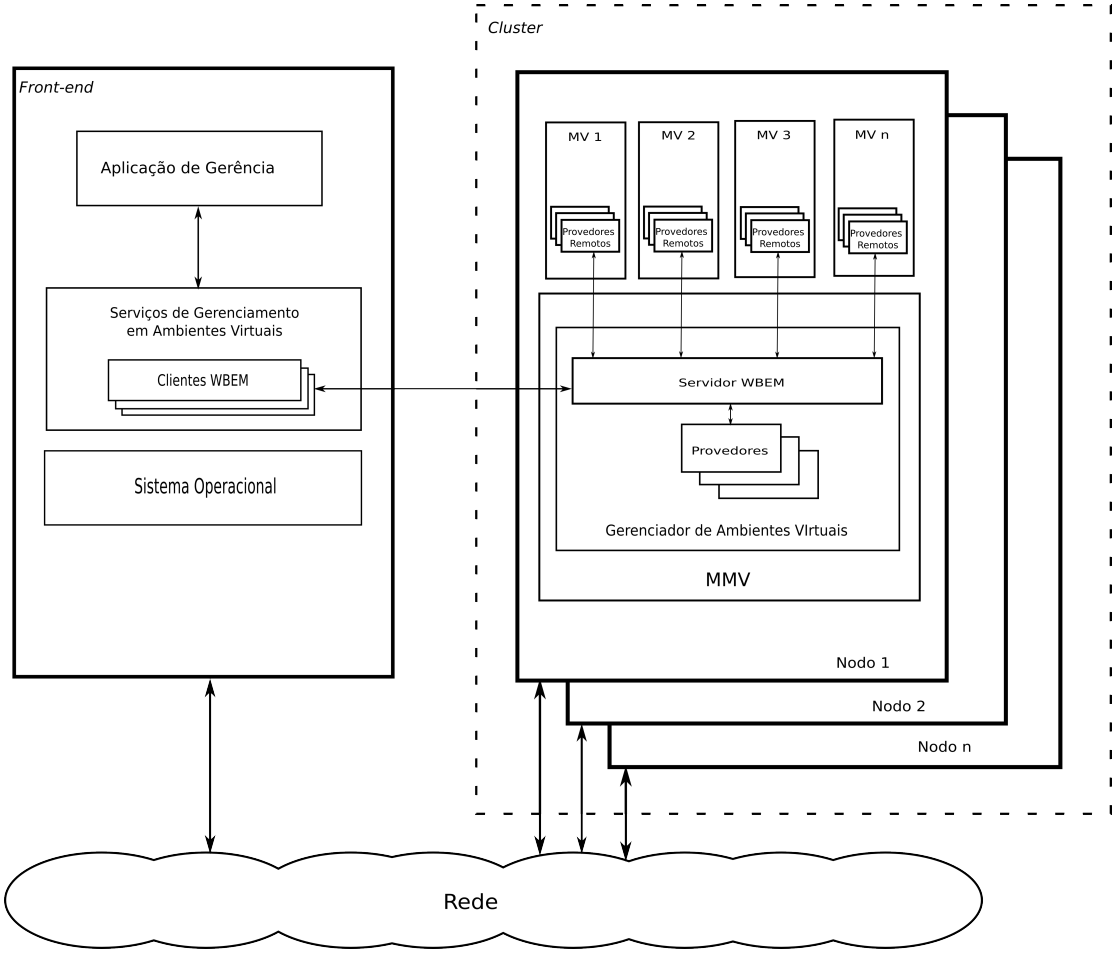

Figura 18: Disposição física dos componentes da arquitetura.

componente utiliza os serviços de monitoração dos recursos das máquinas virtuais para a consulta periódica das informações do ambiente. Ainda, este módulo é responsável por executar serviços que, quando habilitados, monitoram constantemente um determinado recurso do ambiente virtual. Quando este recurso extrapolar os valores determinados, a arquitetura, por meio de alarmes, informa à Aplicação de Gerência a alteração significativa de comportamento do recurso no ambiente. Este componente invoca também serviços que facilitam a gerência do ciclo de vida das máquinas virtuais. Além disso, as aplicações de gerência usam serviços para disparar aplicações executadas sobre o ambiente virtual.

Os Serviços de Gerenciamento em Ambientes Virtuais são um conjunto de serviços invocados pelas aplicações de gerência. Estes serviços estão localizados no *front-end* do *cluster*. Os Serviços de Gerenciamento em Ambientes Virtuais recebem requisições destas aplicações e invocam os Gerenciadores de Ambientes Virtuais, situados no nível privilegiado do virtualizador de cada nodo do *cluster*. A comunicação entre os serviços e os Gerenciadores de Ambientes Virtuais ocorre por meio do padrão WBEM. Cada serviço deste componente possui associado um cliente WBEM, cujo objetivo é realizar requisições aos servidores WBEM presentes em cada Gerenciador de Ambientes Virtuais. Estas requisições podem representar a solicitação de

alguma informação, como a quantidade de CPU utilizada por uma determinada máquina virtual ou a invocação de algum comando, como a criação de uma máquina virtual. Por exemplo, considere que uma aplicação de gerência necessite saber a quantidade de memória usada por uma máquina virtual. Esta aplicação invoca o serviço adequado, informando apenas o identificador da máquina virtual. A seguir, o serviço invocado localiza o nodo que esta máquina virtual se encontra. Após, o serviço invoca o cliente WBEM associado, passando as informações sobre a máquina virtual e o nodo no qual a máquina virtual está hospedada. O cliente WBEM, por sua vez, invoca o servidor WBEM do nodo que a máquina virtual se encontra. Por fim, o servidor WBEM invoca o provedor em questão e repassa as informações requisitadas.

O Gerenciador de Ambientes Virtuais é um componente situado no nível privilegiado do virtualizador de cada nodo do *cluster*. Esta localização permite ao componente executar operações de gerência sobre as máquinas virtuais de um nodo. Este componente recebe requisições do módulo de Serviços de Gerenciamento em Ambientes Virtuais. Cada gerenciador é formado por um servidor WBEM e pelos provedores. Este componente controla os provedores WBEM instalados no nível privilegiado do virtualizador e os provedores remotos, localizados em cada máquina virtual do ambiente. A existência de provedores remotos dentro das máquinas virtuais se deve pela incapacidade dos provedores localizados no nível privilegiado controlarem as aplicações das máquinas virtuais. Assim, os provedores remotos têm a função de monitorar e controlar as aplicações de cada máquina virtual. Os provedores remotos interagem, através do servidor WBEM, com o módulo denominado Gerenciamento de Aplicações. Já os provedores locais interagem, também por meio do servidor WBEM, com os módulos de Gerenciamento de Máquinas Virtuais e de Monitoração de Máquinas Virtuais. Tais módulos são detalhados na próxima seção.

4.2 Detalhamento da Arquitetura

Esta seção apresenta a arquitetura de forma detalhada. Na Figura 19, pode-se perceber a presença de três módulos, que constituem os Serviços de Gerenciamento em Ambientes Virtuais. Além disso, percebe-se a presença dos provedores de monitoração e gerência de máquinas virtuais dentro do nível privilegiado de cada nodo do *cluster*. Já os provedores de aplicação estão situados nas máquinas virtuais do ambiente. Os Serviços de Gerenciamento em Ambientes Virtuais estão organizados em 3 categorias, descritas a seguir:

- Gerenciamento de Máquinas Virtuais: máquinas virtuais não são entidades estáticas. Seu estado pode se modificar de acordo com uma nova necessidade. Máquinas virtuais podem ser movidas de um *host* para outro para otimizar o uso dos recursos computacionais [28]. Desta forma, este módulo engloba serviços relativos ao controle do ciclo de vida das máquinas virtuais. Assim, tal componente permite a realização de tarefas como a cri-

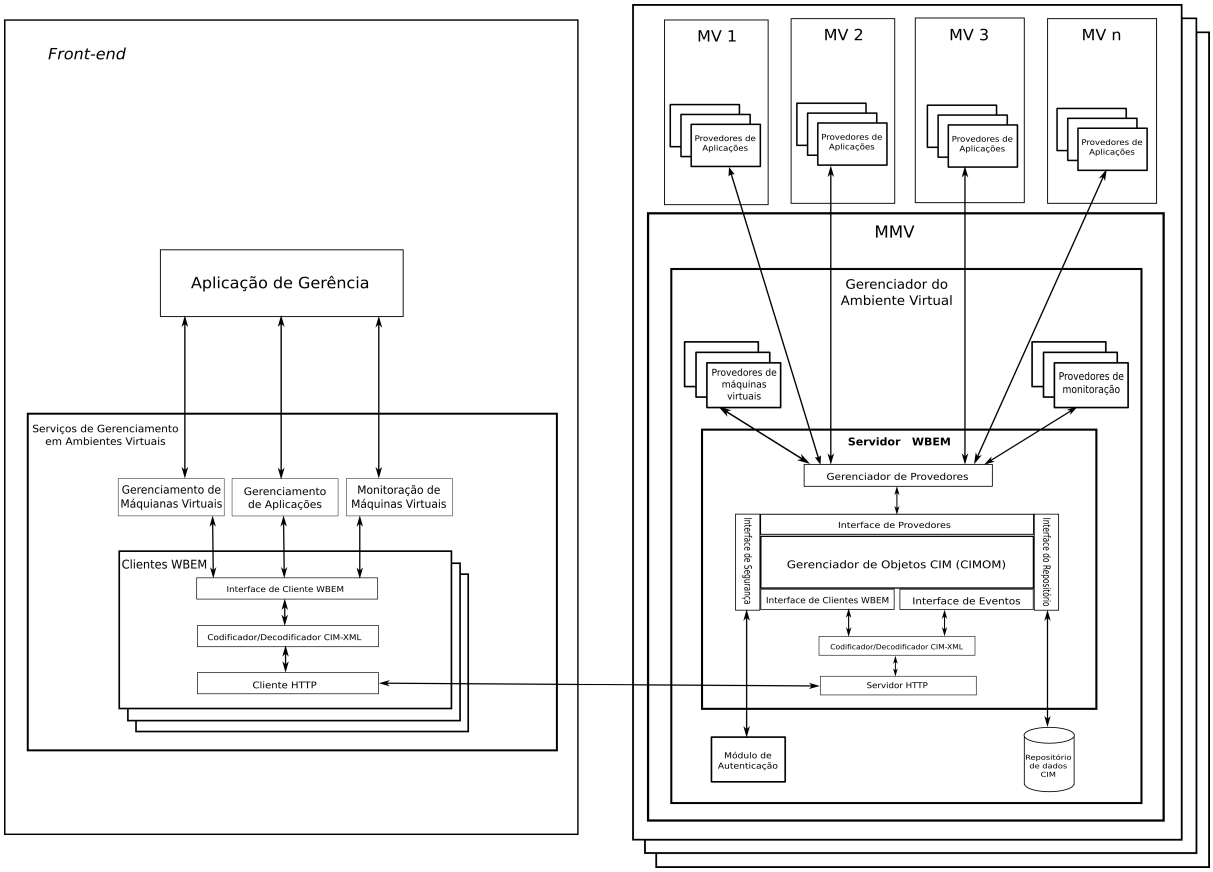

Figura 19: Detalhamento da arquitetura.

ação, destruição, reinicialização, pausa, retomada e migração de uma máquina virtual. Este componente também é responsável por salvar o estado e modificar a quantidade de memória de uma máquina virtual. É importante salientar que durante a criação de uma máquina virtual, as aplicações de gerência podem escolher pela utilização de uma imagem genérica, presente em cada nodo do *cluster* ou uma imagem específica, fornecida pelo usuário.

- Gerenciamento de Aplicações: esse componente fornece serviços para o controle e monitoração das aplicações que executam nas máquinas virtuais do *cluster*. Assim, tal componente é responsável por disparar e finalizar uma aplicação. Por exemplo, considere que uma aplicação de gerência necessite disparar algum processo em um conjunto de máquinas virtuais do ambiente. A aplicação de gerência informa ao serviço apenas o conjunto de identificadores das máquinas virtuais e o caminho do processo no sistema de arquivos da máquina virtual. Em seguida, o serviço descobre a localização de cada máquina virtual e após, através do cliente WBEM associado, inicializa a aplicação desejada em cada máquina virtual. Além da execução de aplicações nas máquinas virtuais, este módulo é responsável também por disponibilizar serviços para a monitoração das aplicações. Dessa forma, este módulo tem capacidade para informar dados como o identificador, o nome,

o estado e a quantidade de memória e de CPU utilizada por cada aplicação. Este módulo é responsável ainda por enviar arquivos de configuração para as máquinas virtuais e resgatar arquivos do ambiente virtual.

- Monitoração de Máquinas Virtuais: esse componente obtém informações de monitoração das máquinas virtuais. Desta forma, o percentual de CPU e a quantidade de memória, utilizados por cada máquina virtual, serão monitorados. Esta monitoração se estende aos parâmetros de rede e espaço em disco de cada máquina virtual. Estas informações são resgatas por meio de *polling*, ou seja, a cada instante que a aplicação necessitar destas informações, o serviço associado à informação deverá ser invocado. No entanto, em algumas situações o gerenciamento proativo é mais adequado. Neste caso, é utilizado o conceito de alarmes. Por exemplo, considere que uma aplicação de gerência necessite saber quando a quantidade de memória usada por uma máquina virtual extrapolar um determinado limite. Para tanto, a aplicação de gerência deve invocar tal serviço, informando o limite e os identificadores das máquinas virtuais que se deseja monitorar. No ambiente virtual, os provedores de indicação monitoram periodicamente o recurso em questão. Quando este recurso extrapolar o limite estabelecido, tal provedor gera um alarme. Este alarme é encaminhado à aplicação de gerência, comunicando-a da alteração do estado do recurso monitorado. Assim, este módulo é responsável por controlar alarmes para a mudança de estado, uso da CPU, utilização da memória e espaço em disco de cada máquina virtual, sendo responsável ainda por informar o nome, o identificador e o estado das máquinas virtuais do ambiente virtual.

Cada um destes módulos apresentados anteriormente estão relacionados com diversos provedores localizados no nível privilegiado e nas máquinas virtuais de cada nodo do *cluster*. Estes provedores são controlados pelo Gerenciador de Ambientes Virtuais. Serviços que monitoram alguma informação possuem provedores de propriedade. Serviços que invocam alguma operação possuem associado provedores de método. Já os alarmes são gerados por provedores de indicação. O Gerenciador de Ambientes Virtuais controla os seguintes provedores:

- Provedores de Máquinas Virtuais: esses provedores, situados no nível privilegiado dos nodos do *cluster*, são invocados para atender os serviços do módulo chamado Gerenciamento de Máquinas Virtuais.

- Provedores de Monitoração e Eventos: localizados no nível privilegiado dos nodos, estes provedores estão associados ao módulo de Monitoração de Máquinas Virtuais. Tais provedores coletam informações das máquinas virtuais. Além disso, os provedores de indicação geram alarmes para os serviços de gerenciamento. Isto acontece quando um recurso extrapolar um limite estabelecido.

- Provedores Remotos de Aplicações: estes provedores estão situados nas máquinas virtuais de cada nodo do *cluster*, já que os provedores localizados no nível privilegiado não

possuem acesso aos processos das máquinas virtuais. Tais provedores atendem às requisições do módulo de Gerenciamento de Aplicações.

4.3 Serviços Disponibilizados pela Arquitetura

Esta seção apresenta de forma detalhada os serviços que a arquitetura disponibiliza para as aplicações de gerência. Estes serviços estão localizados no *front-end* do *cluster* e invocam um ou mais clientes WBEM para realizar suas operações. É importante salientar que a maioria dos serviços apresentados utilizam serviços privados da arquitetura, ou seja, que não são acessíveis pelas aplicações de gerência. Por exemplo, quando uma aplicação deseja reiniciar uma máquina virtual, esta informa apenas o identificador da máquina ao serviço denominado *ReiniciarVM*. A seguir, este serviço invoca outro, chamado *DescobrirHost*. Este último serviço procura o *host* na qual a máquina virtual está situada. Tendo esta informação, o serviço responsável por reiniciar a máquina virtual invoca o servidor WBEM do nodo que a máquina está hospedada. Por sua vez, o servidor WBEM interage com o provedor responsável por reiniciar uma determinada máquina virtual.

O serviço que, a partir de um identificador, encontra a localização das máquinas virtuais é utilizado por quase todos os serviços públicos (que são invocados pelas aplicações de gerência), e por este motivo, não será abordado na descrição de cada um dos serviços apresentados a seguir.

- *CriarVM*: esse serviço é responsável pela criação das máquinas virtuais. Tal serviço recebe como parâmetro a quantidade de memória da máquina virtual e o IP do nodo no qual ela será hospedada. No entanto, antes da criação da máquina virtual, alguns serviços devem ser invocados. Primeiramente, o serviço de DHCP, responsável por atribuir um IP (*Internet Protocol*) à máquina virtual, é invocado. A seguir, um sistema de arquivos para esta nova máquina virtual é criado. Cabe salientar que este serviço é responsável também por manter um identificador único para cada máquina virtual situada no *cluster*, possibilitando que outros serviços da arquitetura localizem uma determinada máquina virtual apenas por meio deste identificador.

- *RecriarVM*: esse serviço permite a recriação de uma máquina virtual. Quando uma máquina virtual é destruída, a imagem que representa o sistema de arquivos usado pela máquina virtual, o seu identificador e o nodo no qual tal máquina virtual estava hospedada são armazenados. Desta forma, caso uma aplicação de gerência necessite utilizar uma máquina virtual destruída, pode-se invocar este serviço de recriação. Deste modo, a máquina virtual é recriada sobre o mesmo sistema de arquivos, identificador e IP. Além disto, a máquina virtual será criada sobre o mesmo nodo do *cluster*.

- *DestruirVM*: esse serviço é responsável por desligar uma máquina virtual. Tal serviço possui algumas variações. No primeiro caso, a aplicação de gerência informa ao serviço uma lista com os identificadores das máquinas virtuais que se deseja destruir. A segunda versão deste serviço permite a destruição de todas as máquinas virtuais do ambiente, ou seja, com apenas uma operação, a aplicação de gerência pode destruir todas as máquinas virtuais do *cluster* virtual. Além disto, o serviço permite que a aplicação de gerência escolha se a imagem que representa o sistema de arquivo das máquinas virtuais será armazenada.

- *ModificarMemoriaVM*: esse serviço permite alterar a quantidade de memória que o virtualizador aloca para as máquinas virtuais. Tal serviço recebe como parâmetro uma estrutura de dados formada pela lista de identificadores das máquinas virtuais e as novas quantidades de memória que serão disponibilizadas para o conjunto de máquinas virtuais. Este serviço, primeiramente, analisa se a máquina virtual está operando. Tal serviço invocará a operação do virtualizador responsável por modificar a quantidade de memória de cada máquina virtual somente se ela estiver ativada.

- *ReiniciarVM*: esse serviço é utilizado para reiniciar as máquinas virtuais. Tal serviço pode reiniciar todas as máquinas virtuais do ambiente virtual ou reiniciar apenas as máquinas indicadas pela aplicação de gerência, através de uma lista de identificadores. No entanto, antes de invocar a operação que reinicializa uma determinada máquina virtual, este serviço verifica se a máquina virtual em questão está ativada.

- *PausarVM*: De forma análoga aos outros serviços, o serviço responsável por pausar uma máquina virtual recebe como parâmetro um conjunto de identificadores de máquinas virtuais. Este serviço verifica também se a máquina virtual em questão está operando. Caso ela esteja ativa, o serviço invoca a operação do virtualizador responsável por pausar a máquina virtual.

- *RetomarVM*: esse serviço é usado para retomar as atividades de uma máquina virtual pausada. O serviço recebe das aplicações de gerência uma lista com os identificadores das máquinas virtuais. No entanto, este serviço utiliza a função do virtualizador responsável por retomar uma máquina virtual apenas se esta estiver pausada. Existe uma estrutura de dados que armazena os identificadores das máquinas virtuais que se encontram no estado pausado.

- *LocalizarVM*: esse serviço, quando invocado por uma aplicação de gerência, informa em qual nodo as máquinas virtuais estão situadas. Tal serviço não recebe nenhum parâmetro. O retorno deste serviço é uma estrutura que armazena, para cada nodo do *cluster*, o nome das máquinas virtuais.

- *RetornarEstadoVM*: esse serviço é utilizado para informar o estado das máquinas virtuais. Os estados podem ser: executando (máquina virtual está utilizando o processador), bloqueada (máquina virtual não está utilizando o processador ou não esta habilitada a utilizá-lo), pausada (máquina virtual utiliza a memória a ela destinada, mas ela não é visível ao escalonador do virtualizador), travada (máquina virtual apresenta algum problema) e desligando (estado anterior à destruição da máquina virtual). A aplicação de gerência deve informar o identificador da máquina virtual. O retorno deste serviço é um destes estados apresentados.

- *RetornarMemóriaUsadaVM*: esse serviço é responsável por informar a quantidade de memória que uma determinada máquina virtual está utilizando. A aplicação de gerência informa como parâmetro o identificador da máquina virtual. O serviço, por sua vez, retorna a quantidade de memória utilizada pela máquina virtual. Cabe salientar que este serviço verifica o estado da máquina virtual em questão antes de efetivamente buscar a informação. Se o estado for diferente de executando, bloqueado ou pausado, a informação não será requisitada pelo serviço.

- *RetornarMemóriaTotalVM*: esse serviço informa às aplicações de gerência a quantidade de memória total designada para cada máquina virtual. Tal serviço primeiramente verifica o estado das máquinas virtuais. Caso estas máquinas estiverem ativadas, isto é, não estiverem travadas ou desligadas, o serviço invoca a operação responsável por retornar a quantidade total de memória da máquina virtual.

- *RetornarCPUUsadaVM*: esse serviço informa o percentual de uso da CPU em um determinado instante. O serviço recebe como parâmetro o identificador da máquina virtual. Como retorno, este serviço informa o percentual de uso da CPU.

- *RetornarMemóriaUsadaHost*: esse serviço, quando invocado por uma aplicação de gerência, informa a quantidade de memória utilizada por um determinado *host*. Como parâmetro de entrada, este serviço recebe o IP do *host* do *cluster*. Antes de buscar a informação, este serviço verifica se o IP informado corresponde a um *host* do ambiente.

- *RetornarCPUUsadaHost*: esse serviço informa o percentual de CPU usado por um determinado *host*. Neste caso, a aplicação de gerência deve informar apenas o IP do *host* a ser monitorado. Analogamente aos outros serviços, esta operação verifica se o IP repassado corresponde a um *host* do *cluster*.

- *RetornarEspaçoDiscoHost*: esse serviço, quando invocado por uma aplicação de gerência, informa o espaço em disco ainda disponível no *host* do *cluster*. Tal aplicação deve informar o IP do *host* a ser monitorado. Este serviço é especialmente útil na fase anterior à criação das máquinas virtuais. Desta forma, a aplicação de gerência verifica se o *host* possui recursos suficientes. Caso este *host* não tenha espaço em disco para hospedar uma

nova máquina virtual, a aplicação de gerência pode decidir por hospedá-la em outro *host*, que possui recursos suficientes.

- *RetornarEspaçoDiscoVM*: esse serviço retorna às aplicações de gerência a quantidade de espaço em disco disponível para as máquinas virtuais. As aplicações informam ao serviço o conjunto de máquinas virtuais que devem ser monitoradas. O serviço, por sua vez, retorna a informação desejada através de uma lista, contendo a quantidade de espaço disponível para cada uma das máquinas virtuais indicadas.

- *MonitorMemoriaUsadaVM*: nesse serviço, as aplicações de gerência devem informar um percentual de memória (limite) e o intervalo de verificação. Em seguida, o serviço invoca, através do servidor WBEM, os provedores de indicação responsáveis por monitorar a quantidade de memória das máquinas virtuais. Este provedor verifica tal informação a cada intervalo de tempo especificado. Quando a informação extrapolar o limite estabelecido, um alarme é enviado à aplicação de gerência. Cabe salientar que este serviço, quando invocado, passa a monitorar a quantidade de memória de todas as máquinas virtuais do ambiente. Quando o alarme é disparado, o serviço envia uma lista com os identificadores de todas as máquinas virtuais do ambiente cujo recurso em questão extrapolou o limite estabelecido.

- *MonitorCPUUsadaVM*: esse serviço, quando invocado, gera alarmes para as aplicações de gerência. Estes alarmes são enviados quando o percentual de CPU das máquinas virtuais extrapolar o limite estabelecido pela aplicação de gerência. Além do percentual de CPU, a aplicação deve informar o intervalo de tempo entre as consultas feitas pelo provedor de indicação. Este serviço informa as máquinas virtuais que extrapolaram o recurso monitorado por meio de uma lista de identificadores de máquinas virtuais.

- *InvocarAplicação*: esse serviço é utilizado para executar ou parar um processo que opera sobre uma determinada máquina virtual. A aplicação de gerência deve informar uma lista com os identificadores das máquinas virtuais e o comando a ser invocado. O serviço recebe estes parâmetros e invoca a aplicação em cada uma das máquinas virtuais indicadas.

- *EnviarArquivosConfiguração*: esse serviço é responsável por enviar arquivos de configuração gerados pelas aplicações de gerência para as máquinas virtuais. A aplicação deve passar como parâmetro a lista de identificadores das máquinas virtuais, a localização do arquivo no sistema de arquivos do *front-end* do *cluster* e o local onde o arquivo será armazenado no sistema de arquivos da máquina virtual. Antes de invocar a operação que envia os arquivos, este serviço verifica se as máquinas virtuais que receberão estes arquivos estão ativadas.

Os serviços apresentados acima são utilizados pelas aplicações de gerência. No capítulo a seguir, tais serviços serão utilizados por uma aplicação de gerência com objetivo de validá-los.

5 Validação da Arquitetura

Este capítulo apresenta a validação da arquitetura descrita anteriormente. Na Seção 5.1, as questões referentes à implementação da arquitetura são avaliadas. Na Seção 5.2, apresenta-se a arquitetura do *framework* para emulação de sistemas distribuídos antes da utilização dos serviços disponibilizados pela Arquitetura de Gerenciamento em Ambientes Virtuais. A Seção 5.3 mostra a interação entre os serviços da arquitetura e o emulador de sistemas distribuídos. Nesta seção, a nova arquitetura do emulador, que passou a se relacionar com os serviços apresentados no Capítulo 4, é apresentada.

Cabe salientar que o *framework* [10] utilizado para a validação deste trabalho está sendo desenvolvido pelo mesmo grupo de pesquisa do trabalho aqui proposto. Este grupo é composto por um aluno de doutorado e dois alunos de mestrado. Além disso, o módulo de Gerência da Rede levou a uma dissertação de mestrado [69].

5.1 Implementação Realizada

Antes de abordar os detalhes relacionados com a elaboração da arquitetura, cabe destacar que a implementação WBEM escolhida foi OpenPegasus, descrita na Seção 3.2.4. O virtualizador escolhido, por questões de projeto, foi o Xen. Suas características podem ser observadas na Seção 2.2.2.

A implementação dos clientes e provedores WBEM foi realizada por meio da linguagem suportada pelo OpenPegasus, C++. No entanto, os serviços disponibilizados para as aplicações de gerência foram implementados em Java, pois a aplicação que utilizou os serviços da arquitetura, o emulador (apresentado na Seção 5.2), usa esta linguagem de programação.

Como visto na Seção 2.2.2, o virtualizador Xen cria um domínio 0 (nível privilegiado) e diversos domínios U (máquinas virtuais). Neste sentido, o servidor WBEM foi instalado no domínio 0 de cada nodo do *cluster*.

Cabe ressaltar que os provedores são invocados pelos serviços de forma indireta. Um serviço, na verdade, invoca um cliente WBEM que por sua vez repassa a requisição para o servidor WBEM de um nodo do *cluster*. Por fim, este servidor WBEM invoca o provedor associado com uma operação, como a reinicialização de uma máquina virtual.

Os provedores invocados (indiretamente) pelos serviços que gerenciam o ciclo de vida das máquinas virtuais interagem com uma aplicação do virtualizador Xen, a Xen Master [15]. Por

meio desta aplicação, o provedor cria, reinicializa, destrói e pausa uma máquina virtual. Além disto, esta aplicação é utilizada para modificar a quantidade de memória de cada máquina virtual.

Já os provedores invocados (de forma indireta) pelos serviços que monitoram as máquinas virtuais interagem com uma biblioteca, a *libxenstat*. Através desta biblioteca, o provedor coleta diversas informações, como a quantidade de memória reservada para cada máquina virtual, o estado destas máquinas virtuais, a taxa de utilização de CPU, as máquinas virtuais hospedadas em cada nodo, etc.

Os alarmes foram implementados por meio de provedores de indicação. Cada um destes provedores possui uma *thread* [53] que, quando disparada, monitora as informações periodicamente. No momento que a informação monitorada extrapolar o limite estabelecido, o provedor envia uma notificação ao cliente WBEM que estava suspenso. As aplicações de gerência que utilizam os serviços de alarme também devem ser implementadas com *threads*. Neste sentido, quando o serviço é invocado, a aplicação de gerência fica suspensa até o recebimento de uma notificação.

Os provedores que disparam as aplicações não utilizam nenhuma biblioteca ou ferramenta para realizar tal tarefa. O serviço, através do cliente WBEM, informa a localização da aplicação no sistema de arquivos da máquina virtual. Após o recebimento deste dado, o provedor invoca a aplicação.

Cada serviço da arquitetura invoca um ou mais clientes WBEM. Por exemplo, o serviço responsável pela criação das máquinas virtuais realiza uma série de operações antes de invocar o provedor, por meio do cliente WBEM, que interage com a aplicação do virtualizador Xen responsável por criar os domínios U. Primeiramente, este serviço invoca o cliente WBEM que informa a quantidade de espaço em disco do nodo no qual se deseja hospedar a máquina virtual. Caso exista espaço, o serviço invoca o cliente WBEM responsável por criar um novo sistema de arquivos (na máquina hospedeira) a partir de uma imagem de sistema de arquivos genérica. A seguir, o serviço cria um identificador global para a máquina virtual e gera o arquivo de configuração usado pelo Xen para criar o novo domínio U. Cabe salientar que estas duas últimas operações são realizadas no *front-end* do *cluster*, e portanto, tais operações não são invocadas por meio de um cliente WBEM. Após, o arquivo de configuração é enviado ao nodo que hospedará a nova máquina virtual. Por fim, o serviço invoca o cliente WBEM responsável por criar a máquina virtual no nodo desejado. Este cliente WBEM, por sua vez, invoca o provedor que interage com a aplicação do Xen, responsável por criar uma nova máquina virtual a partir do arquivo de configuração criado previamente.

5.2 *Framework* para Emulação de Sistemas Distribuídos

O aumento do poder de processamento e o avanço das redes de computadores impulsionaram a utilização de sistemas distribuídos, tanto no meio acadêmico quanto corporativo [19]. Entretanto, as técnicas responsáveis por permitir a avaliação de sistemas distribuídos não avançaram na mesma proporção que a utilização destes sistemas [50]. Neste contexto, está sendo elaborado um *framework* que facilita a avaliação de sistemas distribuídos.

O emulador de sistemas distribuídos utiliza um *cluster* de máquinas virtuais para realizar tal tarefa. As máquinas virtuais são conectadas por meio de um rede virtual, que é configurada de acordo com as necessidades do usuário do *framework*. O mapeamento e a gerência de rede já foram implementados neste projeto. No entanto, este emulador não possui módulos para gerenciar o ciclo de vida das máquinas virtuais, controlar as aplicações que são executadas sobre o *cluster* virtual e monitorar o conjunto de máquinas virtuais do ambiente. Cabe salientar que existe um módulo deste emulador responsável por criar as máquinas virtuais. A seguir, os módulos do emulador são descritos. É importante destacar que esta seção apresenta a arquitetura do *framework* antes da utilização dos serviços disponibilizados pela Arquitetura de Gerenciamento em Ambientes Virtuais.

- Mapeador: um problema recorrente no processo de instalação automatizada de máquinas virtuais é decidir em quais *hosts* de um *cluster* estas máquinas serão hospedadas. Neste sentido, o módulo de mapeamento tem a função de determinar em quais *hosts* do *cluster* as máquinas virtuais serão criadas. Esta definição envolve questões referentes à disponibilidade de recursos nos *hosts*. No mapeamento, o usuário do *framework* deve informar, por meio de dois arquivos XML, a descrição do ambiente real (*hosts* do *cluster*) e a descrição do ambiente requerido. Desta forma, as máquinas que compõem o sistema virtual são mapeadas para os *hosts* que formam o *cluster*. O algoritmo de mapeamento tem dois objetivos principais: o primeiro está relacionado com a minimização do número de *hosts* usados por um usuário, o segundo procura arranjar as máquinas virtuais que se comunicam no *host*. Isto permite reduzir o tráfego de dados pela rede do *cluster*.

- *Deployer*: durante o *deployment*, uma ou mais máquinas virtuais são criadas em cada *host* do *cluster*. No entanto, este módulo tem alguns problemas. Pode-se perceber, por meio da Figura 20, que este módulo invoca instruções do virtualizador, não utilizando nenhum padrão de gerência para realizar tal tarefa. Além disto, o módulo existente não cria um novo sistema de arquivos para cada máquina virtual, não verifica a disponibilidade dos recursos do *host* antes de criar a máquina virtual e não gera um identificador global para a máquina virtual criada. Desta forma, este módulo foi substituído pelos serviços da arquitetura.

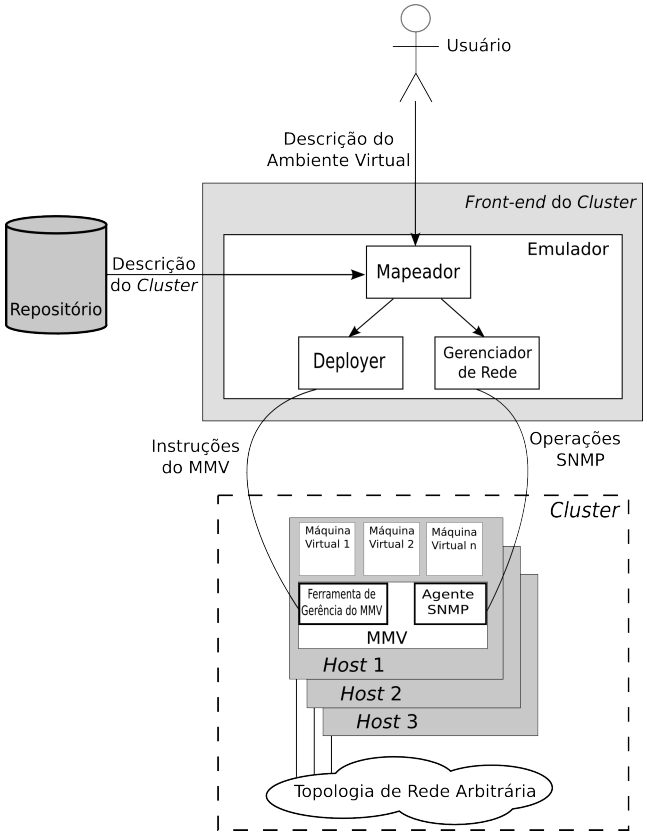

Figura 20: Arquitetura do emulador antes da utilização dos serviços disponibilizados pela Arquitetura de Gerenciamento em Ambientes Virtuais.

- Gerenciador de Rede: após a criação das máquinas virtuais de acordo com o mapeamento realizado, é necessário realizar a conexão entre elas. Desta forma, é possível implantar o sistema emulado a partir das características do sistema distribuído requerido. A partir da descrição do ambiente virtual, este módulo cria as sub-redes. Cada uma destas sub-redes é formada pelas máquinas virtuais situadas nos *hosts* do *cluster*. Após, criam-se conexões entre estas sub-redes através do ajuste da latência e da vazão destas conexões, emulando o comportamento do sistema distribuído requerido. Pode-se perceber, através da Figura 20, que as configurações de rede são realizadas por meio do protocolo SNMP.

A Figura 20 ilustra a arquitetura do emulador de sistemas distribuídos. Note que os componentes desta arquitetura interagem diretamente com o *cluster* virtual. Na próxima seção, será apresentada a arquitetura do emulador após a utilização dos serviços da Arquitetura de Gerenciamento em Ambientes Virtuais.

5.3 Interação entre a Arquitetura e o Emulador de Sistemas Distribuídos

O emulador de sistemas distribuídos, abordado na seção anterior, apresentava alguns problemas. As aplicações eram disparadas manualmente em cada máquina virtual. O módulo de criação de máquinas virtuais não utilizava nenhum padrão de gerência, tornando o emulador específico para o virtualizador Xen, já que o módulo de *deploy* invocava diretamente a ferramenta de gerenciamento do Xen, responsável por criar as máquinas virtuais. Além disto, o emulador não dispunha de mecanismos para realizar tarefas como reiniciar e destruir um conjunto de máquinas virtuais. O emulador não possuía também mecanismos para realizar a monitoração das máquinas virtuais e aplicações que eram executadas no ambiente virtual.

Diante deste contexto, modificou-se a arquitetura do emulador de sistemas distribuídos com o objetivo de utilizar os serviços disponibilizados pela Arquitetura de Gerenciamento em Ambientes Virtuais. Desta forma, o emulador passa a interagir com tais serviços, não se relacionando de forma direta com o *cluster* de máquinas virtuais. A nova arquitetura do emulador é apresentada por meio da Figura 21. Pode-se perceber a presença do Gerenciador de Experimentos, responsável por disparar as aplicações no *cluster* de máquinas virtuais. Percebe-se também o módulo Gerenciador do Ciclo de Vida das Máquinas Virtuais, que utiliza serviços da arquitetura para criar, destruir, reiniciar, pausar e retomar uma máquina virtual. Além disto, foi acrescentado o módulo que monitora as máquinas virtuais, chamado de Monitor. Cabe salientar que a arquitetura não disponibiliza serviços para o módulo chamado Gerenciador de Redes. O Capítulo 6 propõe, por meio dos trabalhos futuros, o desenvolvimento de serviços para este módulo através da especificação WBEM.

É importante destacar que o Mapeador, o Monitor, o Gerenciador do Ciclo de Vida das Máquinas Virtuais, o Gerenciador de Experimentos e o Gerenciador de Rede são módulos do emulador de sistemas distribuídos. Tais módulos apenas utilizam os serviços da Arquitetura de Gerenciamento em Ambientes Virtuais para controlar as máquinas virtuais e as aplicações que operam sobre o ambiente virtualizado. Desta forma, detalhes sobre os algoritmos de mapeamento, sobre os arquivos de descrição do ambiente e sobre a emulação da rede estão fora do escopo deste trabalho.

As próximas subseções mostram a utilização da nova arquitetura do emulador. Primeiramente é apresentada a interação entre os componentes do emulador e os serviços da arquitetura responsáveis pela criação de um ambiente de grade OurGrid [18] automatizado (Subseção 5.3.2). No entanto, antes da apresentação deste experimento, a Subseção 5.3.1 aborda o *middleware OurGrid*. Após, na Subseção 5.3.3, é apresentada a interação entre o emulador e os serviços da arquitetura responsáveis por permitir a adaptação dinâmica do ambiente virtual. Neste último caso, serviços para criação, monitoração, destruição e recriação de máquinas virtuais são utilizados.

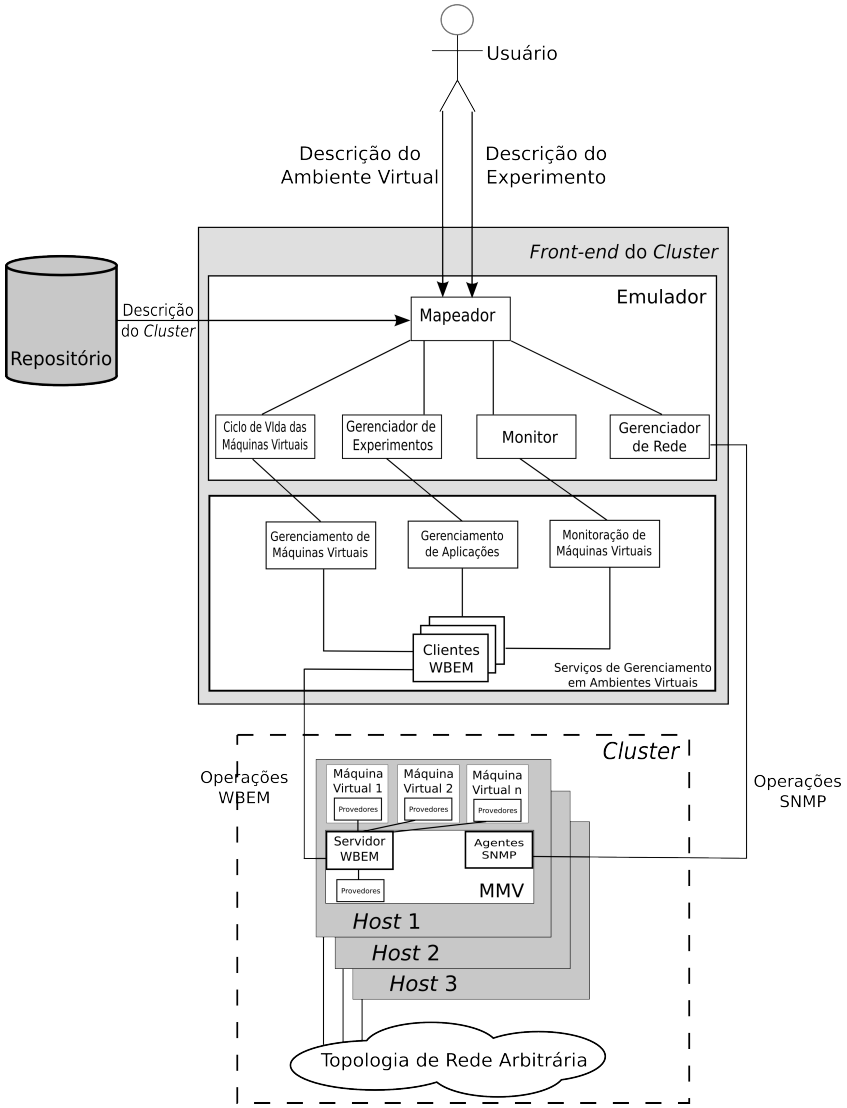

Figura 21: Arquitetura do emulador utilizando os serviços disponibilizados pela Arquitetura de Gerenciamento em Ambientes Virtuais.

5.3.1 OurGrid

OurGrid é um ambiente de grades computacionais desenvolvido pelos pesquisadores da Universidade Federal de Campina Grande.

Segundo Chetty [13], grade computacional pode ser considerada como "*um ambiente paralelo ou distribuído que permite o compartilhamento, seleção e agregação de recursos computacionais distribuídos para resolver problemas de alta escala*".

De acordo com Cirne [18], OurGrid permite o compartilhamento de recursos computacionais ociosos. OurGrid leva em consideração o fato de que as pessoas não utilizam seus computadores o tempo todo. Mesmo quando estes recursos são usados como ferramentas de pesquisa,

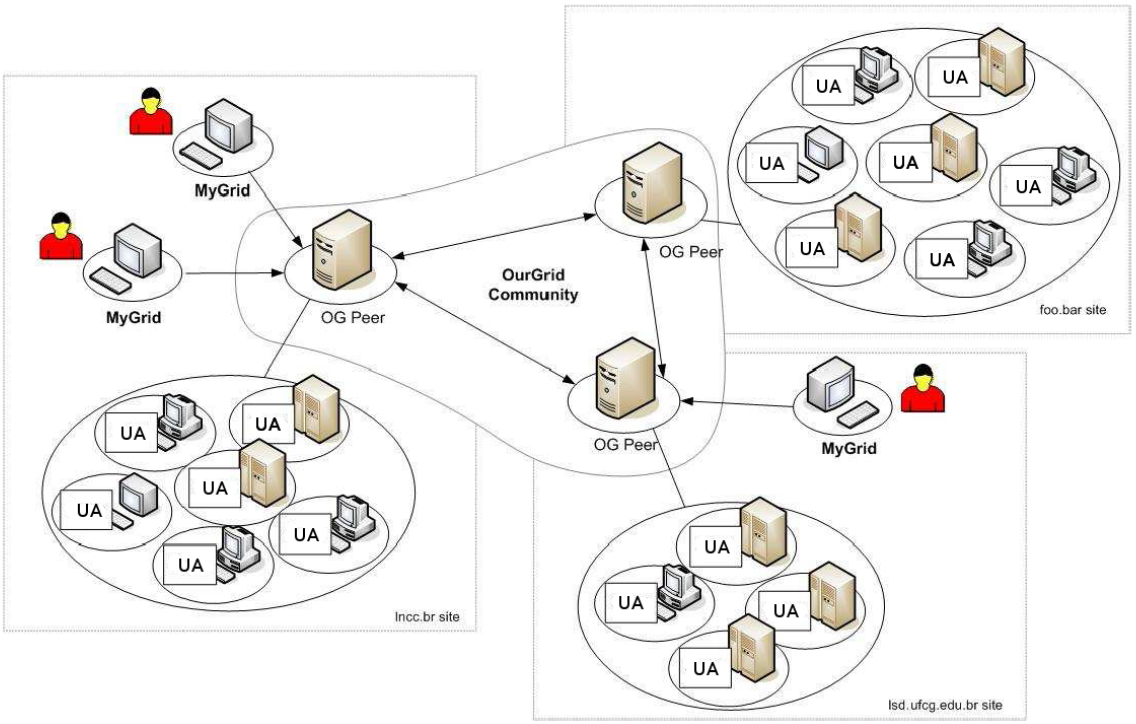

Figura 22: Exemplo de uma grade OurGrid.

pesquisadores alternam entre a execução dos experimentos (que envolve poder computacional) e a análise dos resultados (tempo no qual os recursos permanecem ociosos).

OurGrid é um *middleware* destinado a laboratórios de pesquisa de pequeno ou médio porte que possuem demandas computacionais de grande escala. Os laboratórios disponibilizam seus recursos computacionais ociosos para outros centros de pesquisa por meio do OurGrid [17].

O OurGrid utiliza o conceito de *sites* [16], e suporta o conceito de aplicações BoT (*Bag-of-Tasks*). Estas aplicações são formadas por tarefas independentes, ou seja, não necessitam comunicar umas com as outras [17].

A Figura 22 [18] ilustra um exemplo de uma grade OurGrid. Este ambiente é formado por um conjunto de *sites* que interagem entre si com intuito de obter recursos computacionais. Como se pode perceber por meio da figura, o OurGrid é formado por três componentes principais: MyGrid, Peers e Users Agents (UA) [18].

O MyGrid é o componente do OurGrid que interage com o usuário. Durante o processamento das tarefas, o MyGrid atua como um coordenador, escalonando as tarefas que serão executadas [17]. Estas tarefas são executadas nas máquinas da grade sobre as quais opera o User Agent. Quando um usuário submete tarefas ao MyGrid, este se comunica com o gerente de recursos do domínio, chamado de Peer. Caso não exista máquinas suficientes na grade local para processar as tarefas submetidas ao MyGrid, o Peer local interage com outros *peers*, solicitando seus recursos computacionais disponíveis. O MyGrid normalmente é executado no *desktop* do usuário [18].

Cada domínio possui uma máquina que executa o componente Peer, cuja função é gerenciar as máquinas da grade de cada *site*, permitindo-as, quando ociosas, que processem tarefas de outros *sites*. Cabe salientar ainda que as máquinas da grade, também chamadas de *workers*, executam o componente do OurGrid chamado User Agent [18].

5.3.2 Criação Automatizada de um Ambiente de Grade OurGrid

Esta subseção mostra os detalhes relacionados com a criação automatizada de um ambiente de Grade OurGrid. Para a criação deste cenário, utilizou-se um *cluster* composto por 8 *hosts*. Cada *host* possui um processador Pentium IV 2.8 GHz e 2.560 MB de memória RAM. Tais máquinas são conectadas por um switch Fast Ethernet. Cada *host* possui a plataforma de virtualização Xen instalada. O domínio privilegiado de cada *host* físico possui o servidor WBEM OpenPegasus instalado e sistema operacional Debian GNU/Linux [51].

Quando o emulador é executado, o módulo Mapeador recebe informações sobre o ambiente real (através do arquivo XML de descrição do *cluster*) e sobre o ambiente a ser emulado (por meio do arquivo de descrição do ambiente virtual). A partir destas informações, um algoritmo de mapeamento determina a localização das máquinas virtuais. Neste sentido, o Mapeador invoca o módulo que controla o ciclo de vida das máquinas virtuais, com objetivo de criá-las sobre os *hosts* determinados pelo algoritmo de mapeamento. Após a criação das máquinas virtuais, o Mapeador invoca o módulo responsável por emular o comportamento da rede. Por fim, o Gerenciador de Experimentos, invocado pelo Mapeador, cria o ambiente OurGrid de acordo com o arquivo de descrição do experimento. Este arquivo define quais componentes do OurGrid (Peer, User Agent e MyGrid) serão executados em cada uma das máquinas virtuais do ambiente. Além disto, o arquivo atribui a cada um dos *peers* um conjunto de máquinas da grade, formando os *sites*.

Foi implementado um conjunto de operações específicas para o ambiente Ourgrid que utiliza alguns dos serviços da Arquitetura de Gerenciamento em Ambientes Virtuais, como o serviço de invocação de aplicações e o serviço de envio de arquivos de configuração. Desta forma, o Gerenciador de Experimentos cria o ambiente OurGrid através da invocação destas operações, descritas a seguir:

- *IniciarPeer*: essa operação é responsável por iniciar o componente *peer* nas máquinas virtuais. Ela recebe como parâmetro o identificador da máquina virtual na qual o *peer* será disparado. No OurGrid, antes da inicialização do *peer*, é necessário customizar um arquivo de configuração com informações da máquina virtual que o hospedará. Após a criação deste arquivo de configuração, esta operação envia o arquivo para a máquina virtual desejada. Por fim, esta operação, por meio do serviço de invocação de aplicações, dispara o *peer* na máquina virtual correspondente.

- *IniciarUserAgent*: essa operação tem a função de inicializar o componente User Agent, recebendo como parâmetro o identificador da máquina virtual que o User Agent será disparado. De forma análoga à operação anterior, esta operação cria um arquivo de configuração que será utilizado no momento da invocação do componente User Agent. Depois da criação e do envio deste arquivo para a máquina virtual em questão, a operação inicializa o User Agent através do serviço de invocação de aplicações.

- *IniciarMyGrid*: essa operação dispara o componente MyGrid na máquina virtual desejada por meio do serviço de invocação de aplicações.

- *CriarSite*: essa operação associa um *peer* a um conjunto de máquinas da grade, formando um *site*. A operação recebe como parâmetro o identificador da máquina virtual que hospeda o *peer* e a lista das máquinas virtuais executam o componente User Agent. A partir destas informações, esta operação cria o arquivo de configuração que descreve o *site*, o envia para a máquina virtual que executa o *peer* e invoca (na máquina que o *peer* está sendo executado) o comando responsável por criar o *site* de acordo com o arquivo de configuração.

- *CriarGrade*: essa operação tem a função de associar diversos *peers*, formando uma grade para o componente MyGrid. A operação recebe como parâmetro o identificador da máquina virtual onde o MyGrid está situado e a lista com os identificadores das máquinas virtuais que executam os *peers*. Com tais informações, esta operação cria o arquivo de configuração que descreve a grade para o MyGrid. Sem este arquivo, o MyGrid não tem o conhecimento sobre quais máquinas da grade ele poderá usar para executar as tarefas. Depois da criação do arquivo, esta operação o envia para a máquina que executa o MyGrid e após invoca o comando responsável por carregar o arquivo de descrição da grade.

Existem ainda operações responsáveis por parar os componentes do OurGrid que executam sobre as máquinas virtuais: *PararPeer*, *PararUserAgent* e *PararMyGrid*.

A Figura 23 apresenta um ambiente em grade OurGrid criado sobre o *cluster* virtual de forma automatizada. O ambiente possui 32 máquinas virtuais com 256MB de memória sobre 4 *hosts* físicos, já que o algoritmo de mapeamento minimiza o número de *hosts* de acordo com os recursos disponíveis. O ambiente virtual possui 3 *sites*, que se comunicam por meio de seus *peers*.

5.3.3 Reconfiguração Dinâmica do Ambiente Virtual

Esta subseção mostra o processo de reconfiguração dinâmica do ambiente virtual por meio da utilização dos serviços disponibilizados pela arquitetura. O objetivo do teste realizado é mostrar que os serviços de monitoração detectam a violação no uso de recursos pelas aplicações

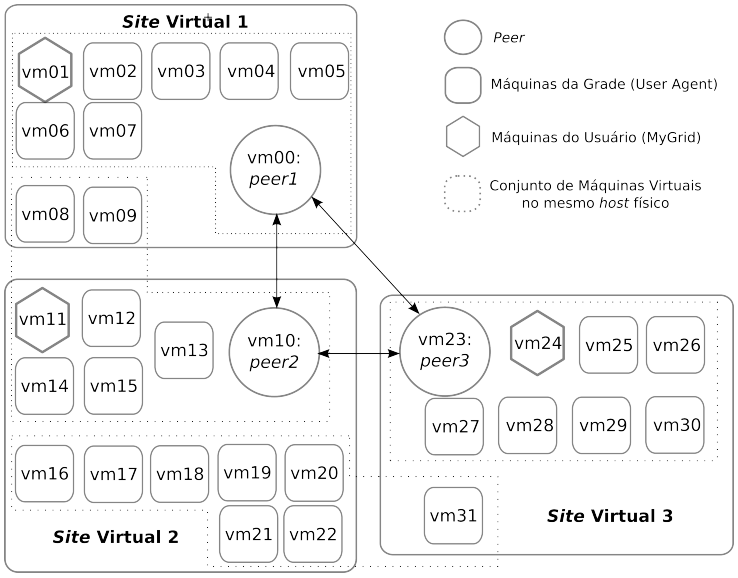

Figura 23: Ambiente construído de forma automática através dos serviços disponibilizados pela arquitetura.

que operam sobre o *cluster* de máquinas virtuais. A partir desta detecção, uma reconfiguração do ambiente é feita por meio da utilização dos serviços que controlam o ciclo de vida das máquinas virtuais da arquitetura.

O ambiente sobre o qual os testes foram realizados consiste em um *cluster* composto por 4 *hosts*. Cada *host* possui um processador Pentium IV 2.8 GHz e 2.560 MB de memória RAM. Tais máquinas são conectadas por um switch Fast Ethernet. Cada *host* possui a plataforma de virtualização Xen instalada. O domínio privilegiado e as máquinas virtuais de cada *host* físico possui o servidor WBEM OpenPegasus instalado e o sistema operacional Debian GNU/Linux.

Cabe salientar que o remapeamento realizado pelo emulador baseia-se na disponibilidade dos recursos do *cluster* e nas condições impostas pelo usuário, através do arquivo de descrição dos experimentos. A partir destas informações, o módulo de mapeamento formula a reconfiguração do ambiente.

O ambiente virtual é formado por 32 máquinas virtuais com 256MB de memória cada, criadas sobre os 4 *hosts* disponíveis. Este ambiente foi construído de forma automática, através dos serviços da arquitetura. Estabeleceu-se, através do arquivo de descrição do experimento, que o ambiente virtual use no mínimo 2 *hosts*. Além disso, foi estabelecido que a quantidade de máquinas virtuais poderá ser diminuída ou aumentada, conforme a utilização dos recursos físicos dos *hosts*. Definiu-se também que as máquinas virtuais não poderão utilizar mais do que 80% da memória a elas disponibilizada.

De acordo com este cenário, utilizou-se uma aplicação responsável por alocar 200MB de memória em cada máquina virtual com o objetivo de forçar a ativação do alarme. Cabe salientar que esta aplicação foi disparada automaticamente em todas as máquinas virtuais por meio dos serviços da arquitetura.

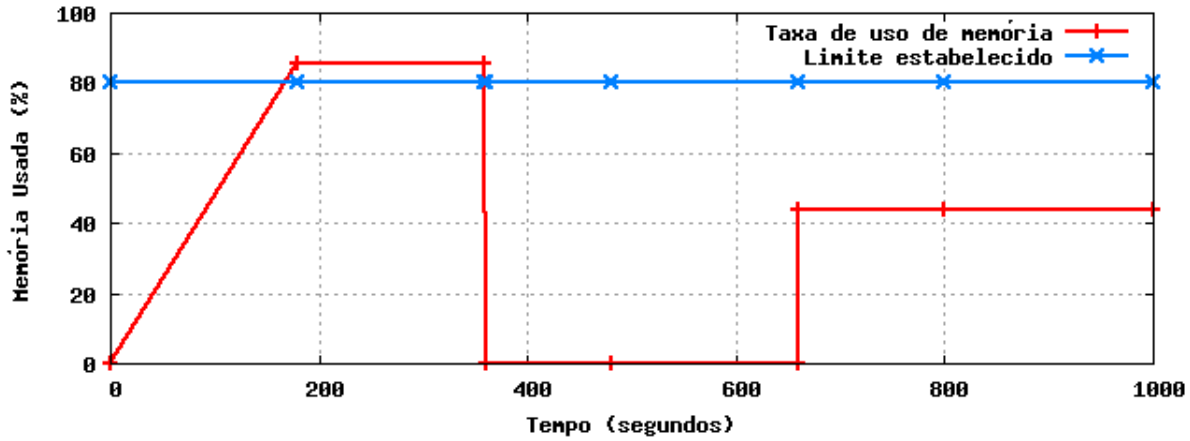

Figura 24: Utilização da memória de uma máquina virtual do ambiente.

A Figura 24 mostra a utilização da memória por uma máquina virtual durante o experimento, enquanto que a Figura 25 apresenta o número de máquinas virtuais rodando no experimento. Analisando o gráfico ilustrado por meio da Figura 24, percebe-se que após 3 minutos, a informação sobre a utilização da memória é coletada. Estes 3 minutos se referem ao tempo que o módulo Monitor demorou para invocar o serviço que retorna a quantidade de memória usada por cada máquina virtual. Depois que a informação é coletada, o módulo Monitor detecta a violação no ambiente virtual e ativa o módulo Mapeador. Baseado na disponibilidade dos recursos do *cluster* e nas condições impostas pelo usuário, é realizado um novo mapeamento do ambiente. A configuração válida encontrada reduziu o número de máquinas virtuais de 32 para 16, como pode ser observado através da Figura 25. Além disso, a configuração dobrou a quantidade de memória de cada máquina virtual. Note, através da Figura 24, que após a reconfiguração do ambiente, aos 11 minutos, a utilização da memória da máquina virtual estabilizou-se em 44%.

Após o Mapeador encontrar uma nova configuração para o ambiente, este módulo invoca o Gerenciador do Ciclo de Vida das Máquinas virtuais, com objetivo de destruir todas as máquinas virtuais do ambiente. A seguir, este módulo é invocado novamente, com objetivo de recriar as máquinas virtuais. Cabe ressaltar que este módulo utiliza os serviços da Arquitetura de Gerenciamento em Ambientes Virtuais para controlar o ciclo de vida das máquinas virtuais. Por fim, a aplicação que consome 200MB de memória de cada máquina virtual é disparada novamente.

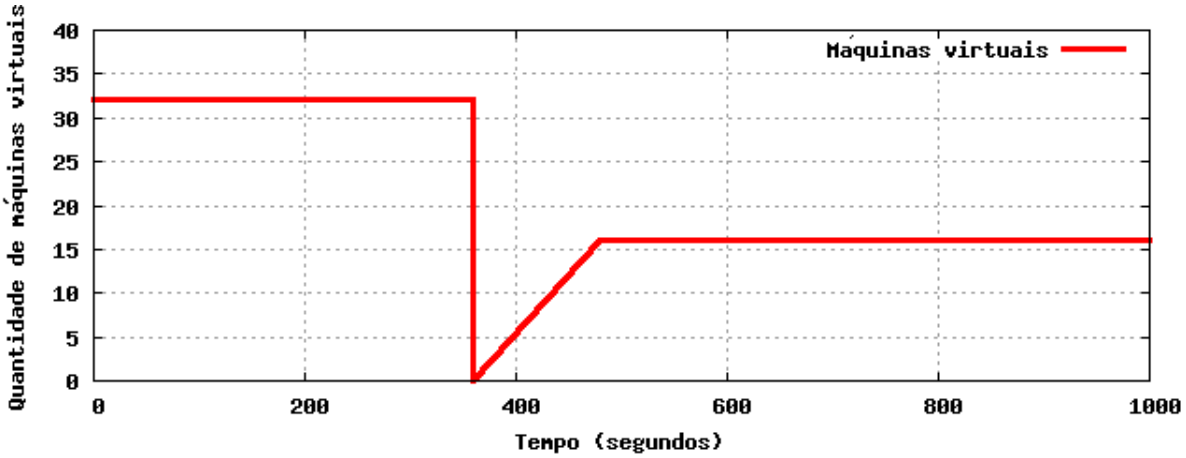

Figura 25: Número de máquinas virtuais no experimento.

6 Considerações Finais

Apesar da utilização de *clusters* de máquinas virtuais por diversas pesquisas no meio acadêmico, as ferramentas existentes para o gerenciamento destes ambientes ainda apresentam problemas. A maioria destas ferramentas preocupa-se apenas em controlar e monitorar máquinas virtuais. Estes trabalhos não se preocupam em gerenciar as aplicações que executam sobre as máquinas virtuais. Além disso, poucos trabalhos disponibilizam serviços que possam ser utilizados por aplicações de gerência. A maioria destes trabalhos tem por objetivo disponibilizar uma interface gráfica para interagir com os administradores do ambiente, impossibilitando a utilização destes trabalhos para o desenvolvimento de aplicações de gerência. Neste sentido, as aplicações que necessitam interagir com um *cluster* de máquinas virtuais precisam implementar mecanismos que não pertencem ao domínio do seu problema, já que o objetivo destas aplicações está relacionado com a emulação de sistemas distribuídos.

Este trabalho apresentou uma arquitetura que se propõe a disponibilizar serviços para a gerência de um *cluster* de máquinas virtuais. Tais serviços permitem controlar o ciclo de vida das máquinas virtuais, monitorar o ambiente virtual e gerenciar as aplicações que operam sobre as máquinas virtuais. Esta arquitetura permite que as aplicações se preocupem apenas na resolução de seus problemas, criando uma camada de abstração que simplifica o gerenciamento do *cluster* de máquinas virtuais.

Como observado no Capítulo 5, esta arquitetura minimizou o problema de indireção causado pela virtualização, pois com a utilização dos serviços desta arquitetura, as aplicações de gerência passam a referenciar apenas as máquinas virtuais. Ainda neste capítulo, foi possível observar a utilização dos serviços da arquitetura por um emulador de sistemas distribuídos. Os serviços permitiram ao emulador a criação de um ambiente de grades OurGrid de forma automática. Além disso, os serviços permitiram que o emulador realizasse a reconfiguração dinâmica do ambiente virtual.

O conjunto de requisitos propostos no Capítulo 4 foi atendido. O Requisito 1 foi atendido através da disponibilização de serviços para as aplicações de gerenciamento, evitando que as aplicações desenvolvam soluções de gerenciamento redundantes e propícias a erros. O Requisito 2 foi atendido na medida que o emulador de sistemas distribuídos, utilizado para a validação da arquitetura, passou a construir o ambiente virtual de forma automática. O Requisito 3 foi atendido, pois os serviços fornecidos pela arquitetura controlam um conjunto de máquinas virtuais com apenas uma operação realizada pelas aplicações de gerenciamento. O Requisito 4 foi garantido através da utilização do padrão WBEM, facilitando a elaboração da arquitetura.

Por fim, o Requisito 5 foi atendido através da utilização de *softwares* de código aberto, como OpenPegasus e Xen.

Existe um componente do emulador, o Gerenciador de Redes que, a partir da especificação do sistema requerido, cria as sub-redes e as conexões entre as máquinas virtuais. Atualmente, o gerenciador realiza as configurações sem a intermediação da arquitetura desenvolvida neste trabalho. Como trabalhos futuros, serão elaborados os serviços responsáveis por emular a rede. Estes serviços deverão ser capazes de isolar as sub-redes e criar as conexões entre as máquinas virtuais, ajustando a latência e a vazão destas conexões. Desta forma, o Gerenciador de Rede passará a relacionar-se com os serviços da arquitetura, não interagindo diretamente com o *cluster* de máquinas virtuais.

Referências

[1] APOSTOLOPOULOS, G.; HASSAPIS, C. V-eM: A cluster of virtual machines for robust, detailed, and high-performance network emulation. In: *14th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems*. Monterey: IEEE Computer Society, 2006. p. 117–126.

[2] BARHAM, P. et al. Xen and the art of virtualization. In: *SOSP '03: Proceedings of the nineteenth ACM symposium on Operating systems principles*. New York: ACM, 2003. p. 164–177.

[3] BELLARD, F. QEMU, a fast and portable dynamic translator. In: *ATEC '05: Proceedings of the annual conference on USENIX Annual Technical Conference*. Berkeley: USENIX Association, 2005. p. 41–41.

[4] BERMUDEZ, A.; CASADO, R. Fast routing computation on Infiniband networks. *IEEE Trans. Parallel Distrib. Syst.*, IEEE Press, Piscataway, v. 17, n. 3, 2006. p. 215–226

[5] BHATIA, N.; VETTER, J. S. Virtual Cluster Management with Xen. In: *Euro-Par Workshops*. Rennes: Springer, 2007. p. 185–194.

[6] BODEN, N. J. et al. Myrinet: A gigabit-per-second local area network. *IEEE Micro*, v. 15, n. 1, 1995. p. 29–36

[7] BOVET, D.; CESATI, M. *Understanding the Linux Kernel*. Sebastopol: O'Reilly & Associates, Inc., 2002. 82p

[8] BUYYA, R. *High Performance Cluster Computing: Architectures and Systems*. Upper Saddle River: Prentice Hall PTR, 1999. 548p

[9] BUYYA, R. *High Performance Cluster Computing: Programming and Applications*. Upper Saddle River: Prentice Hall PTR, 1999. 422p

[10] CALHEIROS, R. et al. Applying virtualization and system management in a cluster to implement an automated emulation testbed for grid applications. In: *SBAC-PAD '08: Proceedings of the 20th Symposium on Computer Architecture and High Performance Computing*. Washington: IEEE Computer Society, 2008. p. 97–104.

[11] CANONICO, R. et al. Virtualization techniques in network emulation systems. In: *Euro-Par 2007 Workshops: Parallel Processing*. Rennes: Springer, 2007. p. 144–153.

[12] CHASE, J. S. et al. Dynamic virtual clusters in a grid site manager. In: *HPDC '03: Proceedings of the 12th IEEE International Symposium on High Performance Distributed Computing*. Washington: IEEE Computer Society, 2003. p. 90.

[13] CHETTY, M.; BUYYA, R. Weaving computational grids: How analogous are they with electrical grids? *Computing in Science and Engineering*, AIP and IEEE Computer Society, v. 4, n. 4, 2002. p. 61–71

[14] CHILDS, S. et al. A virtual TestGrid, or how to replicate a national grid. In: *The 15th IEEE International Symposium on High Performance Distributed Computing*. Paris: IEEE Computer Society, 2006. p. 92–109.

[15] CHISNALL, D. *The Definitive Guide to the Xen Hypervisor (Prentice Hall Open Source Software Development Series)*. Upper Saddle River: Prentice Hall PTR, 2007. 129p

[16] CIRNE, W. et al. Scheduling in bag-of-task grids: The PAUÁ case. In: *SBAC-PAD '04: Proceedings of the 16th Symposium on Computer Architecture and High Performance Computing*. Washington: IEEE Computer Society, 2004. p. 124–131.

[17] CIRNE, W. et al. Running bag-of-tasks applications on computational grids: The MyGrid approach. In: *Proceedings of the 2003 International Conference on Parallel Processing*. Kaohsiung: IEEE Computer Society, 2003. p. 407–416.

[18] CIRNE, W. et al. Labs of the world, unite!!! *Journal of Grid Computing*, Springer, v. 4, n. 3, Sep. 2006. p. 225–246

[19] COULOURIS, G. F.; DOLLIMORE, J. *Distributed systems: concepts and design*. Boston: Addison-Wesley Longman Publishing Co., Inc., 1988. 234p

[20] DE ROSE, C. A. F.; NAVAUX, P. O. A. *Arquiteturas Paralelas*. Porto Alegre: Editora Sagra Luzzatto, 2003. 66p

[21] DEITEL, H. M.; DEITEL, P. J. *C++ How to Program*. Upper Saddle River: Prentice-Hall, Inc., 2000. 178p

[22] DEITEL, H. M.; DEITEL, P. J. *C How to Program: Introducing C++ and Java*. Upper Saddle River: Prentice Hall PTR, 2000. 176p

[23] DEITEL, H. M.; DEITEL, P. J. *Java How to Program*. Upper Saddle River: Prentice Hall PTR, 2001. 391p

[24] DIKE, J. User-Mode Linux. In: *ALS '01: Proceedings of the 5th annual Linux Showcase & Conference*. Berkeley, CA, USA: USENIX Association, 2001. p. 2–2.

[25] DMTF. About DMTF. Disponível em: <http://www.dmtf.org/about>. Capturado em: 14 nov. 2008.

[26] DMTF. CIM Overview. http://www.dmtf.org/cim. Capturado em: 11 oct. 2008.

[27] DMTF. CIM-XML Protocol. Disponível em: <http://www.dmtf.org/standards/wbem/CIM-XML>. Capturado em: 11 nov. 2008.

[28] DMTF. Reducing the Cost and Complexity of Managing a Virtualized Environment). Disponível em: <www.dmtf.org/events/gartner/Virt_Tech_Note_GARTNER.pdf>. Capturado em: 10 out. 2008.

[29] DMTF. Web-Based Enterprise Management. Disponível em: <http://www.dmtf.org/standards/wbem/>. Capturado em: 21 nov. 2008.

[30] DMTF. Web Services for Management. Disponível em: <http://www.dmtf.org/standards/wsman/>. Capturado em: 11 nov. 2008.

[31] FOSTER, I. et al. Virtual clusters for grid communities. In: *CCGRID '06: Proceedings of the Sixth IEEE International Symposium on Cluster Computing and the Grid*. Washington: IEEE Computer Society, 2006. p. 513–520.

[32] GLITHO, R. H. Contrasting osi systems management to SNMP and tmn. *J. Netw. Syst. Manage.*, Plenum Press, New York, v. 6, n. 2, 1998. p. 113–133

[33] GONCALVES, P.; OLIVEIRA, J. L.; AGUIAR, R. L. A WBEM based solution for a 4G network integrated management. In: *ICAS-ICNS '05: Proceedings of the Joint International Conference on Autonomic and Autonomous Systems and International Conference on Networking and Services*. Washington: IEEE Computer Society, 2005. p. 29.

[34] GRALLA, P. *Big Book of Windows Hacks*. Sebastopol: O'Reilly, 2007. 78p

[35] GRAUPNER, S. et al. Impact of virtualization on management systems. In: *HPOVUA 2003: Proceedings of the 10th HP OpenView University Association Workshop*. Geneva: HPOVUA, 2003. p. 10.

[36] HARNEDY, S. J. *Total SNMP: Exploring the Simple Network Management Protocol*. Washington: Cbm Books, 1994. 89p

[37] HEGERING, H.-G.; ABECK, S.; NEUMAIR, B. *Integrated management of networked systems: concepts, architectures, and their operational application*. San Francisco: Morgan Kaufmann Publishers Inc., 1998. p. 89–106

[38] HEILBRONNER, S.; WIES, R. Managing PC networks. *Communications Magazine, IEEE*, v. 35, n. 10, 1997. p. 112–117

[39] HOBBS, C. *A Practical Approach to WBEM/CIM Management*. Boca Raton: CRC Press, Inc., 2004. 154p

[40] HOFFMAN, D.; PRABHAKAR, D.; STROOPER, P. Testing iptables. In: *CASCON '03: Proceedings of the 2003 conference of the Centre for Advanced Studies on Collaborative research*. Toronto: IBM Press, 2003. p. 80–91.

[41] IBM. *Implementing IBM Director 5.20*. Riverton: IBM Corp., 2007. 41p

[42] JIANG, X.; XU, D. SODA: A service-on-demand architecture for application service hosting utility platforms. In: *Proceedings of the 12th IEEE International Symposium on High Performance Distributed Computing*. Seattle: IEEE Computer Society, 2003. p. 174–183.

[43] KAPITONOVA, Y. V. Fundamental ideas and evolution of computer systems. *Cybernetics and Systems Analysis*, Springer, v. 31, n. 2, 1995. p. 218–224

[44] KRSUL, I. et al. Vmplants: Providing and managing virtual machine execution environments for grid computing. In: *SC '04: Proceedings of the 2004 ACM/IEEE conference on Supercomputing*. Washington: IEEE Computer Society, 2004. p. 7.

[45] LEE, S.-J. et al. *Design of a WBEM-based Management System for Ubiquitous Computing Servers*. Disponível em: <www.dmtf.org/education/academicalliance/mjchoi_2004.pdf>. Capturado em: 15 nov. 2008.

[46] LEGRAND, A.; MARCHAL, L.; CASANOVA, H. Scheduling distributed applications: the SimGrid simulation framework. In: *Proceedings of the 3rd IEEE/ACM International Symposium on Cluster Computing and the Grid*. Tokyo: IEEE Computer Society, 2003. p. 138–145.

[47] LEHEY, G.; MCKUSICK, M. K. *The Complete FreeBSD*. Sebastopol: O'Reilly & Associates, Inc., 2003. 248p

[48] MCNETT, M. et al. Usher: An extensible framework for managing custers of virtual machines. In: *LISA'07: Proceedings of the 21st conference on Large Installation System Administration Conference*. Berkeley: USENIX Association, 2007. p. 1–15.

[49] MENASCÉ, D. A. Virtualization: Concepts, applications, and performance modeling. In: *Int. CMG Conference*. Orlando: Computer Measurement Group, 2005. p. 407–414.

[50] MOSLEY, D. J. *Client server software testing on the desktop and the Web*. Upper Saddle River: Prentice Hall PTR, 2000. 328p

[51] MURDOCK, I. Overview of the Debian GNU/Linux system. *Linux J.*, v.6, Oct. 1994. p. 15.

[52] NANDA, S.; CHIUEH, T.-C. *A Survey on Virtualization Technologies*. Department of Computer Science SUNY at Stony Brook, 2005. 15p

[53] NORTHRUP, C. J. *Programming with UNIX Threads*. New York: John Wiley & Sons, Inc., 1996. 85p

[54] OPEN PEGASUS. About OpenPegasus. Disponível em: <http://www.openpegasus.org/>. Capturado em: 13 nov. 2008.

[55] OPEN WBEM. About OpenWBEM. Disponível em: <http://www.openwbem.org/>. Capturado em: 13 nov. 2008.

[56] PARK, J.-G. et al. Cluster management in a virtualized server environment. *Advanced Communication Technology, 2008. ICACT 2008. 10th International Conference on*, v. 3, n. 1, 2008. p. 2211–2214

[57] PLATEAU, B.; ATIF, K. Stochastic automata network of modeling parallel systems. *IEEE Transactions on Software Engineering*, IEEE, v. 17, n. 10, p. 1093–1108, 1991.

[58] PONIATOWSKI, M.; EVANS, W. F. *HP-UX Virtual Partitions*. Upper Saddle River: Prentice Hall PTR, 2002. 27p

[59] RICCI, R. et al. The Flexlab approach to realistic evaluation of networked systems. In: *Proceedings of the Fourth Symposium on Networked Systems Design and Implementation*. Cambridge: USENIX, 2007. p. 201–214.

[60] ROSENBLUM, M. The reincarnation of virtual machines. *Queue*, ACM, New York, v. 2, n. 5, 2004. p. 34–40

[61] SADIKU, M.; OBIOZOR, C. Evolution of computer systems. *Frontiers in Education, Annual*, IEEE Computer Society, Los Alamitos, v. 3, n. 2, 1996. p. 1472–1474

[62] SARMIENTO, E. Securing FreeBSD using Jail. *Sys Admin*, Miller Freeman, Inc., San Francisco, v. 10, n. 5, 2001. p. 31–37

[63] SITES, R. L. *Alpha architecture reference manual*. Newton: Digital Press, 1992. 32p

[64] SMITH, J.; NAIR, R. *Virtual Machines: Versatile Platforms for Systems and Processes (The Morgan Kaufmann Series in Computer Architecture and Design)*. San Francisco: Morgan Kaufmann Publishers Inc., 2005. 340p

[65] SNIA. SMI-S. Disponível em: <http://www.snia.org/forums/smi>. Capturado em: 13 nov. 2008.

[66] SNIA. Storage Networking Industry Association. Disponível em: <http://www.snia.org/>. Capturado em: 13 nov. 2008.

[67] STALLINGS, W. *SNMP,SNMPV2,SNMPV3,and RMON 1 and 2*. Boston: Addison-Wesley Longman Publishing Co., Inc., 1998. 231p

[68] STOKES, J. *Inside the Machine: An Illustrated Introduction to Microprocessors and Computer Architecture*. San Francisco: No Starch Press, 2006. 76p

[69] STORCH, M. *Uma arquitetura de gerência de rede de máquinas com ênfase na emulação de sistemas distribuídos*. DissertaÃ§Ã£o de Mestrado, Programa de Pós-Graduação em Ciência da Computação, PUCRS, 2008. 42p

[70] SUGERMAN, J.; VENKITACHALAM, G.; BENG-HONG, L. Virtualizing I/O devices on VMware workstation's hosted virtual machine monitor. In: *Proceedings of the General Track: 2002 USENIX Annual Technical Conference*. Berkeley: USENIX Association, 2001. p. 1–14.

[71] TAN, Y. an et al. Design and implementation of a wbem disk array provider. In: *PDCAT '05: Proceedings of the Sixth International Conference on Parallel and Distributed Computing Applications and Technologies*. Washington: IEEE Computer Society, 2005. p. 861–865.

[72] TANENBAUM, A. S.; STEEN, M. V. *Distributed systems: Principles and paradigms*. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2001. 204p

[73] THAIN, D.; LIVNY, M. *Building Reliable Clients and Servers*. San Fransisco: Morgan Kaufmann, 2003. 296p

[74] THOMPSON, J. Web-based enterprise management architecture. *Communications Magazine, IEEE*, v. 36, n. 3, Mar 1998. p. 80–86

[75] TUNSTALL, C.; COLE, G. *Developing WMI Solutions*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2002. 52p

[76] VALLÉE, G.; SCOTT, S. L. Xen-OSCAR for cluster virtualization. In: *ISPA Workshops*. Sorrento: Springer, 2006. p. 487–498.

[77] WALTON, S. *Linux Socket Programming*. Indianapolis: Sams, 2001. 36p

[78] WANG, D.; ZHENG, W.; XIONG, J. Research on cluster of workstations. In: *ISPAN '97: Proceedings of the 1997 International Symposium on Parallel Architectures, Algorithms and Networks (ISPAN '97)*. Washington: IEEE Computer Society, 1997. p. 275.

[79] WARD, B. *The Book of VMware*. San Francisco: No Starch Press, 2001. 170p

[80] WBEM SERVICES. About WBEM Services. Disponível em: <http://wbemservices.sourceforge.net/>. Capturado em: 13 nov. 2008.

[81] WBEM SOLUTIONS. Common Information Model Specification. Disponível em: <http://www.wbemsolutions.com/tutorials/DMTF/dmtftutorial.pdf>. Capturado em: 10 nov. 2008.

[82] YOUSEFF, L. et al. *Paravirtualization for HPC Systems*. ISPA, 2006. 42p