Athos Ribeiro de Albuquerque Motta

Migração de servidores de aplicação JAVA no Fundo Nacional de Desenvolvimento da Educação - De Oracle iAS para JBoss

Monografia de Pós-Graduação "*Lato Sensu*" apresentada ao Departamento de Ciência da Computação para obtenção do título de Especialista em "Administração em Redes Linux"

Orientador
Prof. Joaquim Quinteiro Uchoa

Lavras
Minas Gerais - Brasil
2011

Athos Ribeiro de Albuquerque Motta

Migração de servidores de aplicação JAVA no Fundo Nacional de Desenvolvimento da Educação - De Oracle iAS para JBoss

Monografia de Pós-Graduação "*Lato Sensu*" apresentada ao Departamento de Ciência da Computação para obtenção do título de Especialista em "Administração em Redes Linux"

Aprovada em *30 de Abril de 2011*

Prof. Arlindo Follador Neto

Prof. Sanderson Lincohn Gonzaga de Oliveira

Prof. Joaquim Quinteiro Uchoa
(Orientador)

Lavras
Minas Gerais - Brasil
2011

Dedico esta monografia a meus pais, pelo carinho, apoio e incentivo em todos os momentos...

Agradecimentos

Agradeço a Deus pela vida, a minha noiva pelo carinho e companhia, a minha família pelo apoio, e a todos os que me incentivaram durante o curso.

Sumário

Lista de Figuras

Lista de Tabelas

Resumo

O objetivo deste trabalho é apresentar as técnicas e recursos utilizados na migração do ambiente de servidores de aplicações JAVA - J2EE do Fundo Nacional de Desenvolvimento da Educação (FNDE), autarquia do Ministério da Educação. A motivação para efetuar a referida migração foi a atualização para a versão 1.6 do JAVA JDK para desenvolvimento e execução de aplicações, a questão de custos em relação à solução final, a padronização dos ambientes de aplicação com o existente no MEC, além de atender às politicas governamentais de apoio ao *software livre*. O foco principal foi a substituição do servidor de aplicações Oracle iAS 10G para o servidor JBoss EAP 4.3, juntamente com a mudança de equipamentos, implantação de um novo modelo de trabalho dos servidores, fazendo uso de virtualização para aproveitar melhor os novos equipamentos, baseada em Red Hat Linux. Também foram implantados sistemas para monitoramento das aplicações e servidores.

Palavras-Chave: Software Livre; Java; Jboss; Virtualização; KVM; Linux.

Capítulo 1

Introdução

O objetivo deste trabalho é apresentar as técnicas e recursos utilizados na migração do ambiente de servidores de aplicações JAVA - J2EE do Fundo Nacional de Desenvolvimento da Educação (FNDE), autarquia do Ministério da Educação. Após análise de vantagens em relação à estrutura de servidores de aplicação em operação, buscando verificar o que seria mais vantajoso e interessante ao órgão para promover a continuidade de uso da solução existente, foi definido um novo padrão de servidores de aplicação. A motivação para efetuar a referida migração foi a atualização para a versão 1.6 do JAVA JDK para desenvolvimento e execução de aplicações, a questão de custos em relação à solução final, a padronização dos ambientes de aplicação com o existente no MEC, além de atender às politicas governamentais de apoio ao *software* livre. Juntamente com essa mudança de *software* para continuar provendo o serviço de aplicações *web*, foi também efetuada uma substituição de equipamentos por novos servidores com maior capacidade de processamento e com tempo de garantia renovado, permitindo uma maior confiabilidade para os serviços prestados.

Como a diferença de recursos entre os novos equipamentos e os antigos seria grande e de modo a evitar um desperdício desses recursos, decidiu-se utilizar a virtualização de servidores como forma de aumentar a disponibilidade do serviço. Um outro motivo para uso de virtualização era facilitar a manutenção dos servidores e aproveitar de maneira mais satisfatória os recursos disponibilizados por esses equipamentos, tanto para o ambiente de produção como para os ambientes de desenvolvimento e homologação de *software* do FNDE.

Como o sistema operacional utilizado nos equipamentos seria a versão 5.4 do Red Hat Enterprise Linux, foi dada preferência para que se utilizassem os recursos

de virtualização já disponíveis no mesmo, aproveitando as ferramentas ligadas à biblioteca *libvirt*; Havia a possibilidade de utilização de virtualização através de dois hipervisores[1]: o Xen[2] e KVM (Kernel Virtual Machine)[3], ambos sendo *software* de código aberto. Porém, devido a informações recebidas de técnicos da empresa que estaria fornecendo o conjunto de *software* a ser implantado, o KVM seria o produto com suporte principal oferecido pela Red Hat já a partir das novas versões de seu sistema operacional (da versão 6.0 em diante), pelo fato de já estar incluso de forma direta no kernel do Linux e pela grande evolução de sua qualidade desde que foi lançado. Devido a isso, decidiu-se pelo uso dessa solução no ambiente operacional do FNDE.

O foco principal deste trabalho é apresentar a substituição do servidor de aplicações Oracle iAS 10G para o servidor JBoss EAP 4.3 juntamente com a mudança de equipamentos e implantação de um novo modelo de trabalho dos servidores, fazendo uso de virtualização baseada em Red Hat Linux para aproveitar melhor os novos equipamentos. Também foram implantados sistemas para monitoramento das aplicações e servidores.

O texto foi baseado nas normas da UFLA para produção de TCC (PRPG/UFLA, 2007) e encontra-se organizado como se segue. O Capítulo 2 apresenta conceitos básicos sobre ferramentas e recursos utilizados na implantação em estudo, conceitos sobre virtualização com KVM e sobre a utilização da biblioteca libvirt para gerenciamento das máquinas virtuais. O Capítulo 3 apresenta informações sobre o objeto de estudo e como foi realizada a implantação desse estudo no Fundo Nacional de Desenvolvimento da Educação - FNDE. O Capítulo 4 apresenta comentários e observações finais.

[1]Hipervisor: Camada de software que abstrai o *hardware* do sistema operacional permitindo que vários sistemas operacionais virtualizados sejam executados no mesmo equipamento, sendo executado sobre um sistema operacional hospedeiro.

[2]Xen Hypervisor - *http://www.xen.org*

[3]Linux KVM - *http://www.linux-kvm.org*

Capítulo 2

Conceitos Básicos

Este capítulo busca apresentar conceitos básicos relativos aos recursos utilizados na execução do projeto alvo do presente estudo de caso. São apresentadas definições e um breve histórico desses recursos.

Servidores de aplicação são produtos baseados em componentes inseridos na camada central de um ambiente de servidores, provendo serviços de *middleware* para segurança e manutenção de estado, assim como acesso a dados e persistência.

A função de um servidor de aplicação é padronizar a arquitetura de desenvolvimento de aplicações. Esta tarefa é realizada através da definição de vários modelos de componentes, que são padrões que podem ser utilizados pelos desenvolvedores para criar componentes, os quais podem ser disponibilizados em um servidor de aplicação utilizando um modelo de desenvolvimento. A partir do momento em que os componentes são executados no servidor, este provê um conjunto de serviços que são colocados à disposição dos componentes.

Os servidores de aplicação Java baseiam-se na plataforma Java 2, *Enterprise Edition*, que utiliza um modelo distribuído de múltiplas camadas, normalmente sendo formado por:

- uma camada cliente, a qual pode ser composta por uma ou mais aplicações (clientes) ou browsers (no caso de aplicações *web*);
- uma camada central, formada pela plataforma J2EE, através de um servidor *web* e um servidor *Enterprise Java Beans - EJB*, também chamados contêineres;

- uma camada de informações (*Enterprise Information System - EIS*), formada pelos servidores de arquivos e servidores de bancos de dados.

A Figura 2.1 apresenta um exemplo da organização em uma estrutura deste modelo distribuído.

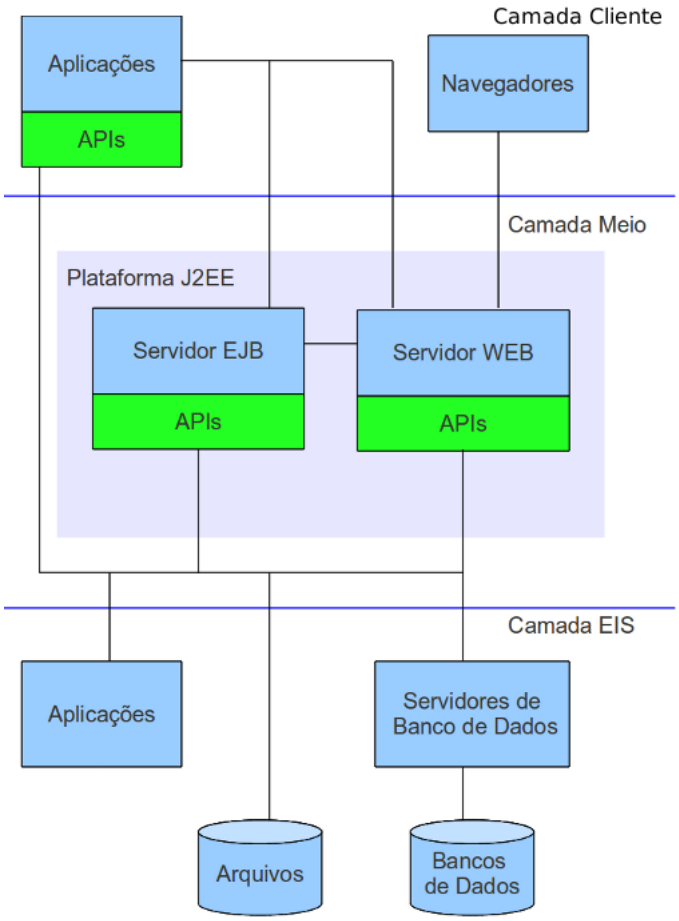

Figura 2.1: Arquitetura J2EE - Adaptada de (BARRY, 2011)

Segundo (ARMSTRONG *et al.*, 2005),

> "Aplicações J2EE de múltiplas camadas são geralmente consideradas como aplicações de três camadas porque elas são distríbuidas entre três localizações: máquinas clientes, o equipamento servidor J2EE,

> e os bancos de dados ou máquinas legadas no *back end*. Aplicações de três camadas que são executadas dessa forma estendem o modelo padrão de camadas cliente e servidor ao inserir um servidor de aplicação de execução múltipla entre a aplicação cliente e o servidor de armazenamento".

A camada cliente pode ser formada por um cliente *web* ou por uma aplicação cliente. O cliente *web* é formado por 2 partes:

- páginas *web* contendo linguagens de marcação[1] (MARKUP..., 2011), dinamicamente geradas por componentes sendo executados na camada *web*;
- um navegador *web*, responsável por renderizar as páginas recebidas do servidor.

Aplicação cliente, mostrada como a camada do topo da Figura 2.1, refere-se ao tipo de cliente disponibilizado através de um navegador web por meio de uma linguagem de marcação. Os componentes da camada do meio, ou camada web são executados no servidor J2EE. Esses componentes são formados por *servlets* ou páginas criadas utilizando tecnologia JSP, disponibilizando conteúdo dinâmico. Por último, a camada EIS, ou sistema de infraestrutura *enterprise*, gerencia os componentes relacionados ao acesso à base de dados e outros sistemas de informação.

2.1 *Oracle Application Server*

O *Oracle Application Server Enterprise Edition* (referido daqui em diante como *Oracle iAS*) é "uma plataforma de aplicações que oferece uma solução compreensiva para desenvolvimento, integração e disponibilização de (...) aplicações, portais e sítios web"[2].

Entre as características citadas pelo fornecedor, estão a alta disponibilidade, performance e escalabilidade. O referido sistema vinha sendo utilizado no ambiente operacional do FNDE aproximadamente desde o ano de 2006, em função de licitação e projeto de implantação ocorridos anteriormente.

[1] Por exemplo: HTML, XML, etc

[2] *Oracle application Server Enterprise Edition* - `http://www.oracle.com/us/products/middleware/application-server/enterprise-edition/index.html`

2.2 Servidor de Aplicações JBoss

O servidor de aplicação Jboss teve origem em 1999, com o nome de EJBoss *(Enterprise Java Beans Open Source Software)*, através de Marc Fleury, disponibilizando uma implementação da porção EJB das especificações da J2EE. O nome foi alterado logo em seguida para evitar problemas de licenciamento com a Sun Microsystems e seu EJB, passando então a se chamar JBoss.

À medida em que o projeto se popularizava, desenvolvedores passaram a vender documentação, serviços de consultoria e treinamento. Em 2001, a equipe de desenvolvimento foi incorporada como JBoss Group, LLC, passando a oferecer serviço de suporte em 2002. Ao mesmo tempo, foi lançada a versão JBoss AS 3, passando a ser competitiva frente a produtos proprietários como Websphere e WebLogic. Em 2004, o JBoss Group, LLC. passou a ser uma corporação com o nome JBoss, Inc. Ao atingir a versão JBoss AS 4, o foco em ambientes corporativos (*enterprise*) aumentou, através de serviços de suporte técnico; Além disso, outras soluções passaram a ser diponibilizadas como componentes que podem ser utilizados tanto junto ao JBoss AS como independentemente. Entre esses componentes, podem ser citados o JBoss Cache, Hibernate, jBPM e JBoss Rules. Em abril de 2006, a empresa foi adquirida pela Red Hat, Inc (JAMAE; JOHNSON, 2009).

Como algumas características do JBoss, pode-se citar a implantação fácil de *cluster* de servidores, com ajustes mínimos em arquivos de configuração e a implantação de novas tecnologias antes dos outros produtos do mercado. Segundo (JAMAE; JOHNSON, 2009), o suporte a Java SE 5.0 e EJB3 no JBoss foi disponibilizado desde 2005, enquanto que os servidores de aplicação de grande influência no mercado passaram a disponibilizar esse suporte apenas dois ou três anos depois.

Além disso, devido ao seu tempo de existência no mercado, o JBoss é considerado um produto maduro e confiável. Por ser um produto desenvolvido no modelo de código aberto (*open source*), sua adoção torna-se mais fácil por parte dos desenvolvedores. O desenvolvedor precisa apenas efetuar o *download* de um arquivo, descompactá-lo, e começar a sua utilização, sem problemas com licenças. Além disso, é possível verificar o código fonte e efetuar modificações se necessário, além de contribuir com o processo de melhorias do JBoss. Da mesma forma que com outro *software open source*, há a formação de uma comunidade de usuários que podem enviar comentários, sugestões, alterações no código através de *patches* enviados e que são analisados para uma possível inclusão em novas versões. Mais ainda, há a possibilidade de se obter suporte gratuito justamente através da comunidade de usuários do sistema.

O servidor de aplicação JBoss é construido em uma arquitetura modular, possibilitando que apenas serviços necessários sejam executados no servidor, evitando assim desperdício de recursos e diminuindo a possibilidade de problemas de segurança devido a processos sendo executados sem necessidade. A instalação do JBoss, como dito anteriormente, pode ser resumida em um simples procedimento de se efetuar o *download* de um arquivo e descompactar o arquivo em um lugar apropriado no equipamento onde o mesmo será executado. Além disso, os arquivos necessários à execução do JBoss ficam todos contidos na estrutura de diretórios provida pelo arquivo descompactado, facilitando a movimentação do serviço em caso de migração de servidores, bem como em caso de necessidade de desinstalação, onde é necessário apenas efetuar a exclusão da referida estrutura de diretórios. A Figura 2.2 ilustra a estrutura de diretórios principais do servidor JBoss (JBOSS, INC., 2006). A Tabela 2.1 apresenta uma descrição dos diretórios superiores da estrutura do JBoss.

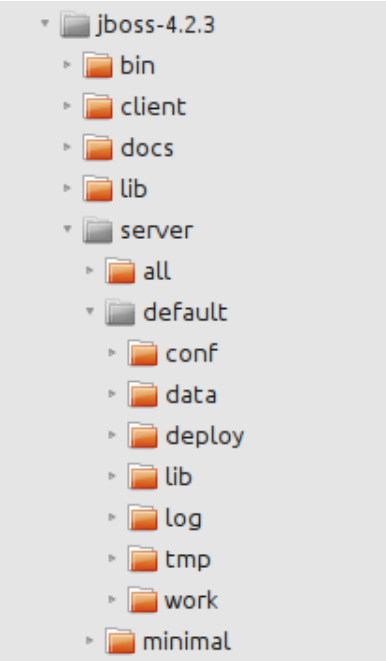

Figura 2.2: Estrutura de diretórios - JBoss

Diretório	**Descrição**
bin	Contém pacotes JAR e scripts de inicialização
client	Contém os pacotes JAR necessários para execução de clientes fora do JBoss
server	Contém os conjuntos de configuração do servidor JBoss
lib	Contém pacotes JAR de inicialização usados pelo JBoss

Tabela 2.1: Estrutura de diretórios superior do JBoss

2.3 Virtualização com KVM

Devido ao aumento constante de recursos e poder de processamento dos computadores, muitas vezes todo o potencial dos equipamentos não é aproveitado. Como forma de resolver essa situação, tem-se dado bastante destaque à utilização de técnicas de virtualização de computadores. A virtualização de sistemas consiste em executar determinado *software*, normalmente sistemas operacionais, de forma isolada, independente e simultaneamente em um único sistema, com sua própria instância de sistema operacional virtualizado. Um dos modos de implementação comuns de virtualização é através da utilização de um hipervisor, que fornece um *hardware* virtual à máquina virtual sendo executada (CURRAN, 2010). A Figura 2.3 apresenta o modelo de camadas de abstração em virtualização.

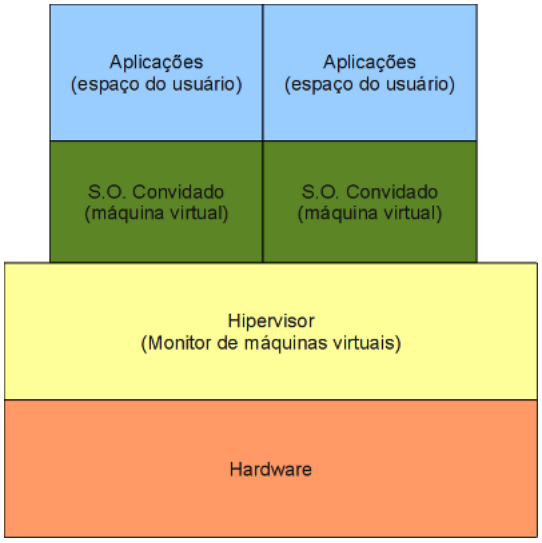

Figura 2.3: Camadas de abstração em virtualização - Adaptado de (JONES, 2007)

O hipervisor serve como uma camada de abstração entre o *hardware* e o sistema operacional. Acima do hipervisor estão os sistemas operacionais "convidados", ou máquinas virtuais. O hipervisor se encarrega de passar a essas máquinas virtuais a impressão de estarem sendo executadas diretamente sobre o *hardware*.

Dentre os modos de virtualização existentes atualmente, pode-se citar:

- **Virtualização por *software*:** Também conhecida como emulação ou virtualização completa, é utilizada para executar sistemas operacionais sem

modificações, apenas simulando toda a arquitetura de CPU do equipamento via *software*. É a mais lenta forma de se virtualizar, porém é a que pode ser executada em mais sistemas, por não depender de *hardware* específico, levando-se em consideração que o sistema de virtualização ofereça a possibilidade de virtualização dessa forma.

- **Tradução binária**: Neste modelo, ao invés de ser efetuada uma emulação do processador, a máquina virtual é executada diretamente na CPU. Caso a máquina virtual necessite ter acesso a instruções privilegidas do sistema, o hipervisor intercepta a chamada e efetua uma reescrita do código em memória, de forma que o sistema operacional da máquina virtual opere normalmente sem perceber a mudança. Este modelo, utilizado pelo hipervisor VMWare[3] (RED HAT, INC., 2009) e também pelo emulador QEMU (BELLARD, 2005), apesar de ter uma alta complexidade para sua implementação, provê ganhos significativos de desempenho em comparação com a emulação completa da CPU.

- **Paravirtualização:** Consiste de uma técnica de virtualização em que o sistema operacional virtualizado é executado juntamente com o sistema hospedeiro, compartilhando dispositivos e bibliotecas comuns a ambos. Um sistema paravirtualizado possui acesso completo aos dispositivos de sistema, sendo que através de configurações de segurança (como *SELinux*[4] e controle de arquivos) é possível impor limites a esse acesso. Uma vantagem da paravirtualização é sua velocidade maior em comparação a outros modos de virtualização, com o custo de ser necessário efetuar modificações no sistema operacional hospedeiro (no caso do *Linux*, uma imagem modificada do *kernel*). O sistema "convidado" fica ciente de que está sendo executado em um hipervisor e coopera com o mesmo em seus processos de escalonamento e I/O; O hipervisor Xen utiliza essa técnica.

- **Virtualização Assistida por hardware:** É uma técnica assistida por *hardware*, provendo um ambiente virtualizado com total abstração da camada física do sistema, criando uma máquina virtual onde o sistema operacional convidado pode ser executado, sem a necessidade de modificações no mesmo. O sistema operacional convidado é executado normalmente, sem a noção de que está em um ambiente virtualizado. É necessário que haja suporte a esse modo de virtualização por parte do processador utilizado, através da presença da tecnologia AMD-V (*AMD Virtualization*) em processadores AMD,

[3] VMWare - `http://www.vmware.com`
[4] SELinux: `http://selinuxproject.org/page/Main_Page`

ou VT-x para processadores Intel; Ambos fabricantes vem constantemente acrescentando aos seus processadores cada vez mais recursos de virtualização de forma a diminuir possíveis sobrecargas à memória do sistema devido à necessidade de tradução de endereços de acesso à memória. Graças a esses avanços, os sistemas virtualizados conseguem atingir desempenho similar a sistemas rodando diretamente sobre o *hardware*. Entre esses recursos estão o VT-D da Intel e o IOMMU da AMD, responsáveis por oferecer acesso direto a dispositivos PCI para as máquinas virtuais. É esse o tipo de virtualização utilizado pelo KVM.

2.4 KVM - *Kernel-Based Virtual Machine*

Os equipamentos atuais têm se tornado cada vez mais complexos em relação a operações de escalonamento básicas, por ser necessário levar em conta múltiplas camadas de processamento em um núcleo, múltiplos núcleos em um processador, e múltiplos processadores em um único sistema. Da mesma forma, as controladoras de memória integradas precisam que o gerenciamento de memória leve em consideração as características de acesso à memória não-uniforme (*NUMA*)[5] do sistema. Enquanto para hipervisores normais é necessário investir esforços para integrar essas capacidades, o *kernel* do Linux apresenta os recursos de escalonador e sistema de gerenciamento de memória de forma madura, sendo assim uma boa opção para virtualização de sistemas, que podem então desfrutar dos trabalhos de otimização atuais e futuros do mesmo em seu favor. Cada máquina virtual passa a ser um processo comum escalonado pelo escalonador do *kernel* Linux.

Nesse contexto é que surgiu o KVM, implementado como um módulo que ao ser carregado converte o *kernel* Linux em um hipervisor. Seu desenvolvimento tem ocorrido de maneira bastante rápida, em parte devido à sua entrada no mercado após tecnologias como a virtualização assistida por *hardware* já haverem sido criadas. Dessa forma, recursos já disponibilizados por *hardware* não precisam ser implementados. O hipervisor KVM passou a fazer parte do *kernel* Linux a partir de sua versão 2.6.20, e requer que o sistema possua suporte às tecnologias Intel VT-X ou AMD-V. Com sua adoção, recursos como gerenciador de memória, escalonador de processos e pilha de rede, entre outros, não precisaram ser redesenvolvidos,

[5]NUMA - *Non-Uniform Memory Access*: Refere-se à uma arquitetura de memória em que o tempo de acesso à memória depende da sua localização em relação ao processador; Dessa forma, o processador pode acessar de forma mais rápida sua própria memória local em comparação a memória não-local, compartilhada ou não com outros processadores.

já que o *kernel* Linux já os tem implantados com uma maturidade e estabilidade reconhecidas.

Ao ter como requisito a disponibilidade por parte do equipamento de recursos nativos de virtualização, o KVM, através de seus desenvolvedores, pode ter um foco voltado a esses equipamentos, ao invés de se preocupar com suporte a equipamentos legados ou modificações no sistema operacional, como no caso do Xen. Além disso, ao utilizar como base o próprio *kernel* Linux, pode se aproveitar dos recursos de gerenciamento de memória, escalonamento de processos, entre outros, já inclusos nele e também da maturidade apresentada pelo mesmo, ao invés de ter que implementar esses recursos desde o início.

As máquinas virtuais são executadas em modo de usuário, sendo gerenciadas pelo sistema operacional hospedeiro como um simples processo. A emulação de dispositivos a serem utilizados pelas máquinas virtuais é de responsabilidade de uma versão modificada do emulador QEMU[6], que fornece um conjunto de dispositivos como controladoras de disco IDE e SCSI, placas de interface de rede, barramentos USB e PCI, entre outros.

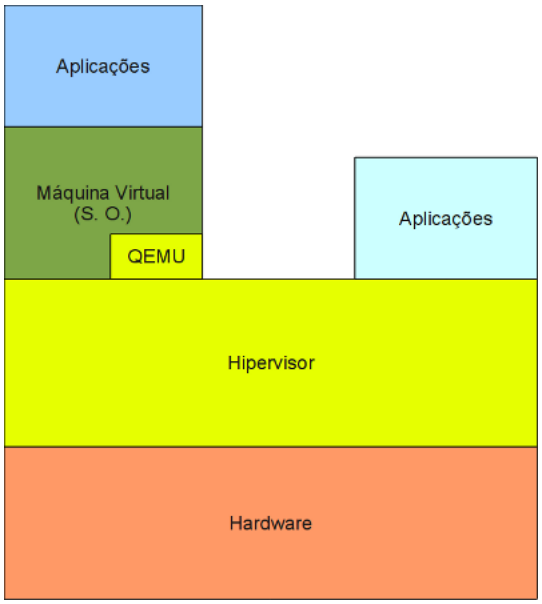

Figura 2.4: Componentes do KVM - Adaptado de (JONES, 2007)

[6]QEMU - http://qemu.org

A Figura 2.4 fornece uma visão do modelo de funcionamento do KVM. Acima do *hardware* está a camada do hipervisor, formada pelo uso do *kernel* Linux juntamente com o módulo do KVM carregado. Acima dele, estão os processos sendo executados pelo sistema operacional, como as aplicações normais do sistema, e juntamente com eles, a máquina virtual junto com suas próprias aplicações, sendo reconhecida também como mais um processo pelo hipervisor (JONES, 2007).

2.5 Libvirt

A biblioteca Libvirt[7] foi desenvolvida para ser um conjunto de *software* destinado a prover uma maneira simplificada de gerenciamento de máquinas virtuais juntamente com outras funcionalidades de virtualização, como gerenciamento de interfaces de rede e armazenamento, ou seja, controlar todos os aspectos de administração de máquinas virtuais. Além disso, um de seus objetivos é prover um meio centralizado de gerência de diversas ferramentas de virtualização, seja hipervisores, sejam outras ferramentas de virtualização. O intuito é ser um "bloco de construção" para ferramentas de gerenciamento de alto nível e para aplicações focadas em virtualização de uma única máquina virtual, ou nó. Diversos nós podem ser acessados simultaneamente, porém as APIs limitam-se a operações em um único nó (LIBVIRT, 2011).

Entre as características da Libvirt, estão a capacidade de gerenciamento remoto dos sistemas virtualizados com utilização de certificados x509[8] e criptografia TLS[9], podendo ser autenticado com Kerberos[10] e SASL[11]; Ela permite uso de políticas de controle de acesso utilizando PolicyKit[12] e autodescoberta de sistemas através de DNS *multicast* com Avahi[13]. O desenvolvimento da biblioteca está voltado a oferecer uma API comum que implemente ofereça funcionalidades comuns

[7] Libvirt: http://libvirt.org

[8] x509: Padrão de infraestrutura de chaves publicas para ponto único de acesso (*Single sign-on(SSO)* e infraestrutura de gerenciamento de privilégios(PMI): http://tools.ietf.org/html/rfc2510

[9] TLS - Segurança em camada de transporte(*Transport Layer Security*: http://www.ietf.org/html.charters/tls-charter.html

[10] Kerberos - Protocolo de autenticação de rede: http://web.mit.edu/Kerberos/

[11] SASL - Camada simples de autenticação e segurança(*Simple Security and Authentication Layer*): http://asg.web.cmu.edu/sasl/

[12] PolicyKit - Conjunto de ferramentas para controle de privilégios para serviços do sistema. http://www.freedesktop.org/wiki/Software/PolicyKit

[13] http://avahi.org/

Tabela 2.2: Hipervisores suportados pela Libvirt

Hipervisor	**Descrição**
Xen	Hipervisor para arquiteturas Intel 32bit, Intel 64bit e PowerPc 970
QEMU	Emulador disponível para várias arquiteturas
Kernel-based Virtual Machine (KVM)	Plataforma de emulação Linux
OpenVZ	Virtualização em nível de sistema operacional baseada no Kernel Linux
VirtualBox	Hipervisor para virtualização de arquitetura x86
User Mode Linux	Emulador de plataforma Linux para várias arquiteturas
Armazenamento	Drivers de áreas de armazenamento (disco local, disco em rede, volumes iSCSI)

implementadas pelo conjunto de hipervisores suportados. A Tabela 2.2 mostra uma lista de hipervisores suportados pela libvirt.

2.6 Recursos de administração

2.6.1 Virsh

A terminologia utilizada pela Libvirt refere-se ao *host* físico como sendo um "nó" e ao sistema operacional virtualizado como sendo um "domínio". A configuração de um domínio é toda definida através de um arquivo XML, contendo as opções necessárias para definição do domínio, informando recursos utilizados pelo domínio, configuração de periféricos, tipos de dispositivos do domínio. Essa configuração pode ser muito simples, ao mesmo tempo em que permite um grande detalhamento de características disponibilizadas para o dominio em questão.

Uma das ferramentas disponibilizadas pela Libvirt chama-se Virsh. Trata-se de um comando que permite a utilização de funcionalidades da Libvirt através de um modelo interativo de *shell*. Os recursos oferecidos por essa ferramentas podem ser utilizados diretamente pela linha de comando, e também pelo shell interativo invocado pela execução isolada do comando *virsh*.

Através desse comando, é possível efetuar operações como:

Tabela 2.3: Exemplos de parâmetros do comando virsh.

Parâmetro	**Função**
autostart	Habilita e desabilita a inicialização automática de um domínio ao se iniciar o serviço (*daemon* da libvirt.
console	Efetua conexão à console virtual da máquina virtual (domínio).
create	Cria um domínio a partir de um arquivo XML.
destroy	Finaliza imediatamente a execução de um domínio, liberando qualquer recurso que o mesmo esteja utilizando.
detach-disk	Desconecta um dispositivo de disco do domínio.
domifstat	Obtém informações referentes à interface de rede de um domínio em execução.
dominfo	Obtém informações básicas sobre determinado domínio.
dumpxml	Retorna informações sobre o domínio em formato XML na saída padrão (*stdout*).
edit	Possibilita a edição do arquivo XML de configuração de um domínio.
list	Retorna uma lista de domínio configurados e/ou disponíveis.
migrate	Efetua a migração de um domínio para outro *host*.
reboot	Efetua a reinicialização de um domínio.
start	Coloca um domínio desligado em funcionamento.

- listar as máquinas virtuais, ou domínios, existentes;
- repassar comandos para iniciar, reiniciar e desligar os domínios;
- alterar configurações do domínio, seja informando novos dispositivos como redimensionando recursos;
- efetuar migrações de domínios entre hosts.

A Tabela 2.3 mostra algumas das opções disponíveis à ferramenta, que podem ser utilizadas tanto como parâmetros em linha de comando (por exemplo, “virsh list”), ou diretamente ao se utilizar o *shell virsh*.

2.6.2 Virtual Machine Manager

Um dos recursos disponíveis no RedHat e em outras distribuições atuais para fazer o gerenciamento de máquinas virtuais é o *Virtual Machine Manager*[14], ou Virt-manager. O *Virtual Machine Manager* é uma ferramenta em interface gráfica que disponibiliza ao administrador recursos como um sumário dos domínios sendo executados, juntamente com gráficos exibindo sua utilização de memória e processamento. Ainda oferece ferramentas de auxílio (*wizards*) para criação, configuração e alterações em domínios, ajudando na tarefa de administrar os recursos de *hardware* virtual para os mesmos, além de fornecer acesso a um console gráfico através de um cliente VNC embutido. Juntamente com o Virt-manager, são disponibilizadas algumas ferramentas para auxiliar o trabalho do administrador. Essas ferramentas, executadas em modo de linha de comando, são:

- **Virt-install**: Oferece meios de fazer a instalação de sistemas operacionais nos domínios;
- **Virt-clone**: Para efetuar cópias de domínios inativos;
- **Virt-image**: Para efetuar instalações de sistemas operacionais a partir de uma imagem pré-definida.
- **Virt-viewer**: Fornece um console gráfico do sistema operacional convidado.

A Figura 2.5 apresenta a tela inicial do *Virtual Machine Manager*, já com máquinas virtuais criadas e em execução. Cabe salientar que os nomes dos servidores foram alterados na figura de modo a preservar informações referentes ao ambiente existente. Nesta tela é possível verificar informações como a quantidade de memória alocada para cada máquina virtual, bem como o servidor (*host*) onde as máquinas estão sendo executadas. Outra informação disponível refere-se à quantidade de processadores e/ou núcleos disponíveis no *host* e à quantidade de processadores virtuais alocados para cada máquina virtual.

A Figura 2.6 exibe informações sobre o uso de recursos que uma determinada máquina virtual está consumindo em tempo real (uso de processador, uso de memória, tráfego de dados em disco e de rede).

[14] Virtual Machine Manager - `http://virt-manager.et.redhat.com/index.html`

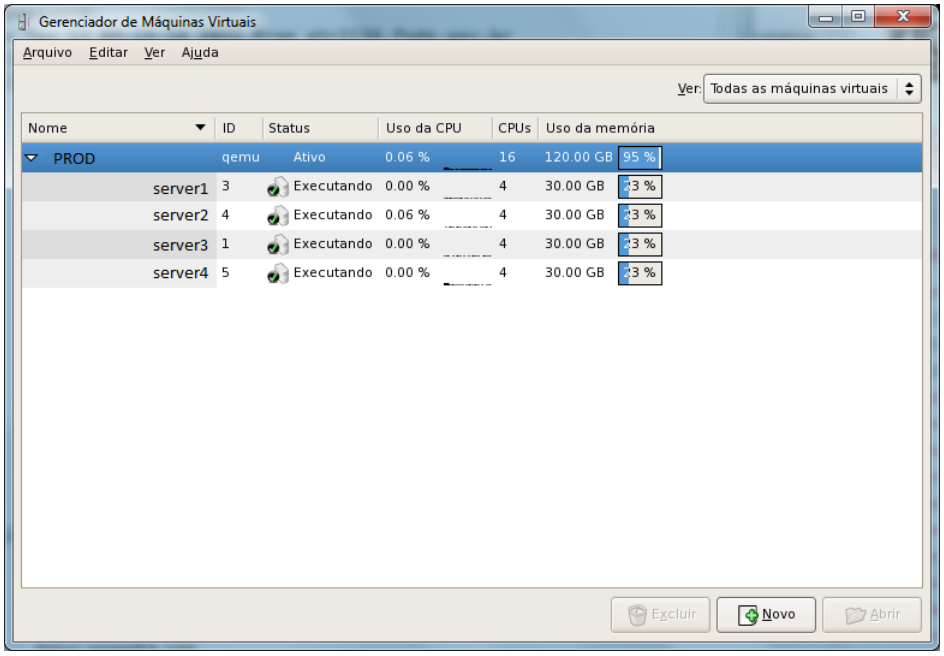

Figura 2.5: *Virtual Machine Manager* - Tela principal

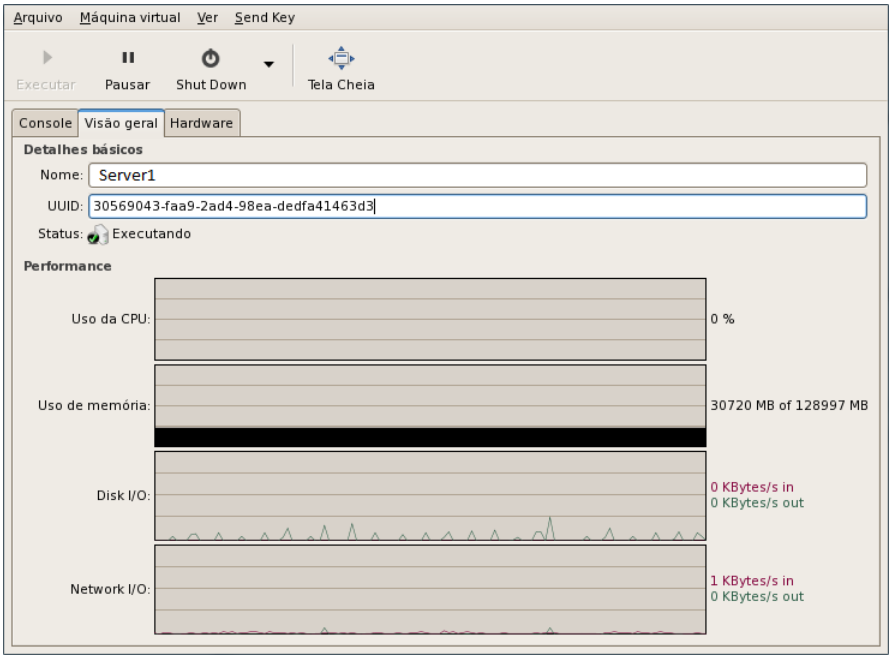

Figura 2.6: *Virtual Machine Manager* - Informações de máquina virtual

Capítulo 3

Estudo de Caso

3.1 Ambiente de estudo

O ambiente de aplicações JAVA do FNDE era composto por um servidor dedicado para o desenvolvimento das aplicações, um servidor para se efetuar a homologação dos sistemas desenvolvidos antes de entrarem em produção, e o ambiente de produção em si, formado por quatro servidores físicos que comunicavam entre si através de um sistema de arquivos em cluster, o OCFS2[1]. As especificações dos equipamentos são listadas na tabela 3.1.

O acesso aos sistemas em produção é controlado por um equipamento balanceador de carga, sendo que o utilizado é o *appliance* netscaler, de fabricação da Citrix. Esse balanceador é responsável por toda a parte de gerenciamento de contextos das aplicações e redirecionamento do usuário a determinado servidor dentro de um conjunto. Esse equipamento mantem-se em uso no ambiente, sofrendo alterações apenas nos servidores para os quais passou a direcionar o tráfego da rede.

O ambiente posterior passou a ser formado também por seis máquinas físicas, sendo quatro para o ambiente de produção, uma para desenvolvimento e mais uma para aplicação. A diferença primária está nas especificações de *hardware* desses equipamentos, como listadas na tabela 3.2.

A mudança de ambiente em relação à alteração de sistema operacional ocorreu de forma tranquila, uma vez que ambos, tanto SUSE como Red Hat utilizam como base o padrão de pacotes RPM. Dessa forma, houve um melhor aproveitamento

[1] OCFS2 - *Oracle Cluster File System 2*: `http://oss.oracle.com/projects/ocfs2/`

Tabela 3.1: Especificações dos equipamentos - ambiente anterior

Desenvolvimento:	Um servidor, composto por: • um processador Intel Xeon de núcleo duplo com tecnologia *Hyperthreading*, resultando em quatro núcleos visíveis para o sistema; • 6GB de memória; • sistema operacional SUSE Linux Enterprise Server 10 SP2.
Homologação:	Um servidor, composto por: • um processador Intel Xeon de núcleo duplo com tecnologia *Hyperthreading*, resultando em quatro núcleos visíveis para o sistema; • 8GB de memória; • sistema operacional SUSE Linux Enterprise Server 10 SP2.
Produção:	Quatro servidores, compostos por: • quatro processadores Intel Xeon de núcleo duplo com tecnologia *Hyperthreading*, resultando em dezesseis núcleos visíveis para o sistema; • 12GB de memória; • sistema operacional SUSE Linux Enterprise Server 10 SP2.

Tabela 3.2: Especificações dos equipamentos - novo ambiente

Desenvolvimento:	Um servidor, composto por: • dois processadores Intel Xeon de núcleo quádruplo, resultando em oito núcleos visíveis para o sistema; • 32GB de memória; • sistema operacional Red Hat Enterprise Linux 5.4.
Homologação:	Um servidor, composto por: • dois processadores Intel Xeon de núcleo quádruplo, resultando em oito núcleos visíveis para o sistema; • 32GB de memória; • sistema operacional Red Hat Enterprise Linux 5.4.
Produção:	quatro servidores, compostos por: • quatro processadores Intel Xeon de núcleo quádruplo, resultando em dezesseis núcleos visíveis para o sistema; • 128GB de memória; • sistema operacional Red Hat Enterprise Linux 5.4.

do tempo, já que os responsáveis pela administração dos servidores já possuíam bastante familiaridade com a estrutura e o funcionamento do sistema operacional. Para aproveitar recursos de gerenciamento e evitar tráfego desnecessário na rede, foi configurado um servidor como repositório de pacotes para distribuição nos servidores.

A implantação teve início com o levantamento por parte dos técnicos da empresa contratada das configurações do ambiente de aplicações existente no FNDE, através de diversas reuniões e acompanhamento dos administradores do ambiente. Foram obtidas informações sobre as bibliotecas utilizadas, os sistemas que estavam sendo disponibilizados, o modo de funcionamento destes sistemas, as fontes de dados para conexão ao banco de dados, e como as aplicações eram disponibilizadas ao servidor de aplicação durante suas devidas atualizações e implantações.

O primeiro passo foi o levantamento de quais aplicações seriam portadas do ambiente antigo para o Jboss. Após esse levantamento, foi verificado que aproximadamente trinta sistemas deveriam ser migrados ao todo. Porém, foi verificado que alguns sistemas precisariam continuar funcionando em um ambiente servido pelo Oracle Application Server, devido à esses sistemas serem desenvolvidos em linguagem *PL/SQL*, dependente do ambiente Oracle. Assim não seria possível efetuar a migração de imediato desses sistemas. Dessa forma, ficou a cargo dos desenvolvedores e arquitetos JAVA a tarefa de efetuar posteriormente o trabalho de desenvolvimento de solução independente de ambiente caso possível e necessário a médio ou longo prazo.

Um dos desafios enfrentados foi a mudança das bibliotecas de sistema que por vezes encontravam-se diretamente atreladas às aplicações em seus empacotamentos, os quais eram disponibilizados através dos pacotes .ear específicos de cada aplicação. Durante o processo de migração, essas bibliotecas foram redefinidas de modo a serem utilizadas em modo compartilhado pelas aplicações, mantendo assim um ambiente mais controlado e com risco menor de falhas.

De modo a não causar transtornos aos usuários dos sistemas, foram preparados ambientes separados nas novas máquinas enquanto os servidores antigos continuavam em produção. Dessa forma, o processo de migração tanto de bibliotecas para garantir compatibilidade com o JBoss quanto das aplicações pode ser feito de forma gradual e segura. À medida que as aplicações iam sendo migradas, os responsáveis pela homologação dos sistemas efetuavam os testes no novo ambiente e validavam os mesmos para implantação em produção.

Enquanto isso, o desenvolvimento de novas versões das aplicações foi congelado no ambiente antigo, e os desenvolvedores passaram a efetuar alterações já

visando o novo ambiente. Ao mesmo tempo, os servidores destinados a prover o novo ambiente de produção iam sendo instalados e configurados para hospedar o serviço do JBoss juntamente com as aplicações portadas.

Foram criados em cada servidor físico de produção quatro máquinas virtuais, além de duas máquinas virtuais no servidor físico de desenvolvimento e duas no servidor de homologação, utilizando-se a virtualização do KVM juntamente com o uso da biblioteca libvirt para gerenciamento das máquinas virtuais.

Inicialmente, cogitou-se utilizar uma área de armazenamento comum em um *storage* para esse ambiente. Devido a problemas de performance detectados após testes utilizando GFS[2], decidiu-se manter as máquinas independentes entre si, porém mantendo as mesmas versões das aplicações em instalações iguais. Isto foi feito através da utilização da ferramenta Rsync[3] de uma das máquinas virtuais para todas as outras, garantindo a igualdade de versões, e com o balanceamento de carga garantido pelo equipamento também já citado anteriormente. Apenas para informação, o motivo dos problemas de desempenho foi devido a um gargalo existente no referido equipamento de armazenamento.

Através da utilização das máquinas nessa disposição, passou-se a ter um ambiente com maior capacidade de conexões por parte dos usuários dos sistemas. Além disso um problema que às vezes acontecia com o sistema anterior foi eliminado. Os servidores rodando o Oracle iAS em determinados momentos não obtiam êxito na sincronização. Os servidores não conseguiam atualizar os sistemas através dos pacotes .ear e também não conseguiam voltar ao funcionamento normal após esses procedimentos de atualização, deixando as aplicações indisponíveis em alguns momentos. Isso exigia a parada dos processos responsáveis pelos sistemas em todas as máquinas do ambiente. Em alguns casos, era necessário efetuar a reinicialização dos servidores e até mesmo a restauração de cópias de segurança.

Outro ponto alterado foi a maneira como a implantação e atualização de sistemas era efetuada. No ambiente anterior, as aplicações eram disponibilizadas a um dos servidores de aplicação, e através do sistema de arquivos de *cluster* OCFS2, os dados eram replicados para as outras máquinas responsáveis pelo mesmo trabalho. Tentou-se utilizar GFS para que as máquinas enxergassem um ponto de acesso único através de um storage. Porém, como o desempenho não ficou como o esperado, decidiu-se utilizar os antigos *scripts* adaptados para o novo ambiente, replicando as mudanças para todos os servidores através do uso da ferramenta Rsync. Assim, mesmo com o aumento do número de servidores, que passaram

[2]GFS - *Red Hat Global File System*: `http://www.redhat.com/gfs/`
[3]Rsync: `http://rsync.samba.org/`

de quatro para dezesseis no ambiente de produção, o tempo de atualização das aplicações foi otimizado.

O *script* de atualização das aplicações tem seu funcionamento baseado na utilização de um menu em *shell script* que apresenta ao responsável pela atualização do sistema a opção de escolha do sistema a ser atualizado. O *script* obtém um arquivo contendo os códigos da aplicação disponibilizado em um servidor de arquivos compartilhado, copia para o servidor de aplicação, efetua as devidas alterações de arquivos de configuração de aplicações que necessitam manter-se constantes entre atualizações, e então através do uso de Rsync, efetua apenas as alterações mais atuais no sistema, melhorando o tempo de atualização.

Assim que as aplicações migradas nos ambientes de homologação e desenvolvimento foram definidas como prontas para implantação no ambiente de produção, foi efetuado um plano de migração através de padrões de requisição de mudanças. As aplicações foram separadas em uma lista e organizadas para migração por etapas, após análise de impactos e revisões por parte dos desenvolvedores e gerentes responsáveis pelo funcionamento das mesmas. A migração foi dividida em três grupos para migração, levando em consideração níveis de complexidade e de prioridades de funcionamento dessas aplicações. Foram identificados alguns erros de execução das aplicações, os quais foram corrigidos para efetuar a implantação final. Ao final da implantação, devido a alguns problemas de funcionamento de duas aplicações, houve a necessidade de se efetuar o retorno de sua execução para o ambiente anterior até que os problemas fossem solucionados, e após novos testes, houve a migração final com sucesso.

De forma a possibilitar o monitoramento dos servidores e serviços disponibilizados pelos mesmos, foi efetuada configuração por meio de adição desses novos servidores a ferramentas de monitoramento como Nagios e Cacti de modo a garantir ações por parte dos técnicos em caso de problemas. Como houve a distribuição do serviço entre vários servidores, o padrão de disponibilidade dos serviços prestados passou a manter-se em um nível maior que o anterior.

Capítulo 4

Conclusão

A motivação principal do objeto desse estudo foi de se efetuar a migração dos servidores de aplicação JAVA e ao mesmo tempo atualizar os equipamentos que disponibilizavam os serviços anteriormente. Os equipamentos em utilização ainda apresentavam bom desempenho e cumpriam suas funções, porém devido ao fato de término de seus prazos de garantia e também à necessidade de se manter o funcionamento tanto dos equipamentos como dos serviços prestados por eles é que houve essa migração.

Uma das grandes deficiências do ambiente Java / *Oracle iAS* no FNDE era a questão da versão do ambiente de desenvolvimento e execução de aplicações Java que vinha sendo utilizada, mais especificamente com a versão 1.4 do *Java Development Kit - JDK*, defasada e apresentando menos recursos em relação às versões mais atuais. A dificuldade de atualização estava principalmente na necessidade de instalação de versão mais atual do sistema, sendo necessário firmar novo contrato junto à Oracle (na figura de seus representante comerciais), causando um elevado custo devido ao alto preço de licenças de uso de software desta empresa.

Como o objetivo principal da migração era efetuar a transferência das aplicações do ambiente com Oracle iAS para o novo ambiente executando JBoss, é possível dizer que a mesma foi efetuada com sucesso. Após as alterações necessárias devido às diferenças de funcionamento dos dois produtos, os sistemas foram migrados e passaram a trabalhar como esperado no novo ambiente. Todo o processo foi feito seguindo recomendações de práticas de gerência de informação como por exemplo documentos de requisição de mudança. Dessa forma, apenas com a verificação de funcionamento dos sistemas é que o trabalho foi dado como concluido. Passado um ano após a migração ter sido efetivada, a solução implan-

tada baseada no servidor de aplicações JBoss tem se mostrado estável, confiável e de ótimo nível de qualidade. Exceto por problemas isolados em certos momentos provenientes de alterações em aplicações, o serviço tem sido mantido sem necessidade de grandes alterações ou incidentes críticos. Além disso, a qualidade dos produtos utilizados tem sido verificada em função de seu funcionamento durante esse tempo de uso.

Uma das questões que puderam ser observadas é a confirmação de que *software* livre nem sempre é sinônimo de *software* gratuito. O contrato de suporte estabelecido junto à empresa, e por sua vez junto à Red Hat, foi de 1 ano, sendo que após expirado esse prazo, houve a necessidade de se renovar a subscrição dos serviços de suporte. Durante a transição de período de vencimento de suporte e sua renovação, houve a interrupção da possibilidade de se obter novos pacotes e atualizações através dos canais oficiais da Red Hat. Além disso, pode-se observar que há um custo de certa forma elevado em alguns pontos, como em relação ao suporte de sistema operacional, mas ainda assim vantajoso em relação à solução utilizada anteriormente.

Referências Bibliográficas

ARMSTRONG, E.; BALL, J.; BODOFF, S.; CARSON, D. B.; EVANS, I.; GREEN, D.; HAASE, K.; JENDROCK, E. *The J2EE 1.4 Tutorial*. 2005. 1542 p. Disponível em: <http://download.oracle.com/javaee/1.4/tutorial/doc-/J2EETutorial.pdf>.

BARRY, D. Application server definition. 2011. Acessado em 20/02/2011. Disponível em: <http://www.service-architecture.com/application-servers-/articles/application_server_definition.html>.

BELLARD, F. Qemu, a fast and portable dynamic translator. *Proceedings of the USENIX Annual Technical Conference*, Usenix Association, p. 6, 2005. Disponível em: <http://www.usenix.org/event/usenix05/tech/freenix/full_papers-/bellard/bellard.pdf>.

CURRAN, C. *Red Hat Enterprise Linux Virtualization Guide*. 5. ed. [S.l.], 2010. 402 p. Disponível em: <http://docs.redhat.com/docs/en-US-/Red%5FHat%5FEnterprise%5FLinux/5/html/Virtualization>.

JAMAE, J.; JOHNSON, P. *JBoss in Action*. Greenwich, CT: Manning Publications Co., 2009. 464 p.

JBOSS, INC. *The JBoss 4 Installation Guide*. [S.l.], 2006. 26 p. P.14,15. Disponível em: <http://docs.jboss.org/jbossas/guides/installguide/r1/en/pdf-/jboss4-install.pdf>.

JONES, M. T. Discover the linux kernel virtual machine. *IBM DeveloperWorks*, IBM Corp., 2007. Acessado em 20/02/2011. Disponível em: <http://www.ibm-.com/developerworks/linux/library/l-linux-kvm/>.

LIBVIRT. *Libvirt Goals*. 2011. Online. Disponível em: <http://libvirt.org/goals-.html>.

MARKUP Language. *Britannica Online Encyclopedia*, Britannica Online Encyclopedia, 2011. Acessado em 14/03/2011. Disponível em: <http://www-.britannica.com/EBchecked/topic% -/1323641/markup-language>.

PRPG/UFLA. *Normas para Elaboração de Trabalhos de Conclusão de Cursos de Pós-Graduação Lato Sensu*. 2. ed. Lavras, 2007. 29 p. Disponível em: <http://www.prpg.ufla.br/Legis/legis1.htm>.

RED HAT, INC. *KVM – KERNEL BASED VIRTUAL MACHINE*. [S.l.], 2009. 11 p. Disponível em: <http://www.redhat.com/f/pdf/rhev/DOC-KVM.pdf>.