UNIVERSIDADE SAGRADO CORAÇÃO

EMANUEL JOSÉ LARINI MELI

FRAMEWORK WEB PARA DESENVOLVIMENTO DE ALGORITMOS DE ALOCAÇÃO DE MÁQUINAS VIRTUAIS EM SERVIDORES NA COMPUTAÇÃO EM NUVEM

BAURU
2015

EMANUEL JOSÉ LARINI MELI

FRAMEWORK WEB PARA DESENVOLVIMENTO DE ALGORITMOS DE ALOCAÇÃO DE MÁQUINAS VIRTUAIS EM SERVIDORES NA COMPUTAÇÃO EM NUVEM

Trabalho de Conclusão de Curso apresentado ao Centro de Ciências Exatas e Sociais Aplicadas como parte dos requisitos para obtenção do Título de Bacharel em Ciência da Computação, sob orientação do Prof. Me. Henrique Pachioni Martins.

BAURU
2015

EMANUEL JOSÉ LARINI MELI

FRAMEWORK WEB PARA DESENVOLVIMENTO DE ALGORITMOS DE ALOCAÇÃO DE MÁQUINAS VIRTUAIS EM SERVIDORES NA COMPUTAÇÃO EM NUVEM

Trabalho de Conclusão de Curso apresentado ao Centro de Ciências Exatas e Sociais Aplicadas da Universidade Sagrado Coração como parte dos requisitos para obtenção do Título de Bacharel em Ciência da Computação sob orientação do Prof. Me. Henrique Pachioni Martins.

Banca examinadora:

_____

Prof. Me. Henrique Pachioni Martins

Universidade Sagrado Coração

_____

Prof. Me. Patrick Pedreira Silva
Universidade Sagrado Coração

_____

Prof. Esp. André Luiz Ferraz Castro

Universidade Sagrado Coração

Meli, Emanuel Jose Larini
M522f

Framework Web para o desenvolvimento de algoritmos de alocação de máquinas virtuais em servidores na computação em nuvem / Emanuel Jose Larini Meli. -- 2015.
54f. : il.

Orientador: Prof. Me. Henrique Pachioni Martins.

Trabalho de Conclusão de Curso (Graduação em Ciência da Computação) – Universidade do Sagrado Coração – Bauru – SP.

1. Computação em nuvem. 2. Ferramenta Web. 3. Algoritmos de Alocação. 4. PHP. 5. JavaScript. I. Martins, Henrique Pachioni. II. Título.

RESUMO

A Computação em Nuvem é um paradigma que fornece infraestrutura, plataforma de desenvolvimento e aplicativos em formato de serviços sob demanda e cobrados conforme o uso. Na prática, a Nuvem é implementada em Datacenters, onde se concentram servidores físicos que executam máquinas virtuais através das quais são prestados os serviços de computação. Para que seja garantida a qualidade destes serviços, é necessário o uso de técnicas. Uma dessas técnicas é a alocação de Máquinas Virtuais em Datacenters. Este trabalho teve como principal objetivo desenvolver uma ferramenta Web para testes de algoritmos de alocação de Máquinas Virtuais em Servidores pertencentes a um Datancenter, considerando somente uma dimensão. A metodologia utilizada consiste em pesquisas bibliográficas, projeto e implementação das classes nas linguagens PHP e Javascript utilizando os conceitos de padrões de projeto MVC. Para validação foi utilizado um algoritmo de alocação e sua demonstração de execução na ferramenta, bem como seus testes. Através da validação comprova-se o funcionamento da ferramenta, bem como sua utilização no paradigma da alocação de Máquinas Virtuais em Hosts na Nuvem.

Palavras-Chave: Computação em nuvem. Ferramenta Web. MVC. Algoritmos de Alocação. PHP. Javascript.

ABSTRACT

Cloud Computing is a paradigm which provides infrastructure, development platform and applications in form of on demand services and charged under usage. In practice, the Cloud is implemented in Datacenters having physical Hosts which execute Virtual Machines. To ensure the quality of its services is necessary some techniques. One of them is the VM allocation policy. This Project aimed to develop a Web framework considering only one dimensiono of a VM. The metodology used consists in bibliographic search, code build and design in PHP and Javascript using the concepts of MVC project. To validate the framework has been used one of literature VM allocation pollicy, its demonstration in framework and tests. Through validation it was possible to prove the framework efficiency and also its utilization in Cloud allocation pollicy paradigm.

Keywords: Cloud Computing. Web Framework. MVC. VM Allocation. PHP. Javascript.

LISTA DE FIGURAS

LISTA DE SIGLAS

AJAX – Asynchronous Javascript and XML

CRM – Customer Relationship Manager

CSS – Cascade Stylesheet

GPL – General Public License

HTML – Hypertet Markup Language

IaaS – Infrastructure as a Service

MIPS – Milhões de Instruções por Segundo

MMV – Monitorador de Máquina Virtual

MV – Máquina Virtual

MVC – Model-View-Control

NIST – National Institute of Standards and Technology

PaaS – Platform as a Service

PHP – Hypertext Preprocessor

SaaS – Software as a Service

SO – Sistema Operacional

TI – Tecnologia da Informação

SUMÁRIO

1 INTRODUÇÃO

Os primeiros computadores surgiram na década de 40 durante a Segunda Guerra Mundial. Desde então, crescente avanço no setor de Tecnologia e Informação (TI) fez com que surgissem novos recursos das mais variadas formas e utilidades.

Uma dessas novas tecnologias, a Computação em Nuvem, se estabeleceu nos últimos anos como uma plataforma de grandes vantagens que pretende prover serviços a todos, desde o usuário final às empresas que terceirizam todo seu setor de TI para outras empresas. Dessa forma, usuários podem mover seus dados e aplicações diretamente para uma nuvem, acessando-os de forma simples e eficaz.

Segundo Taurion (2009), esse termo surgiu em 2006 numa palestra de Eric Schmidt, sobre o método utilizado pela sua empresa para gerenciar seus datacenters[1]. A possibilidade de armazenamento e processamento de dados em um servidor na Web, através dos serviços oferecidos pela Computação em Nuvem, permite que uma grande diversidade de dispositivos (tablets, celulares, notebooks e desktops), possa acessar e executar esses recursos, sendo necessário somente o acesso à internet e um mecanismo padronizado, que por sua vez pode ser um navegador que necessita poucos recursos computacionais.

Sabahi (2011) define Computação em Nuvem como sendo um ambiente de rede baseado no compartilhamento de recursos computacionais. Os usuários finais nem mesmo se dão conta de que estão trabalhando com aplicações em nuvens, como por exemplo, o Google Drive[2]. Na computação em nuvem é oferecido tanto hardware como software nos centros de dados. Empresas que fornecem nuvens utilizam tecnologias de virtualização combinadas com suas habilidades para disponibilizar recursos de computação através de sua infraestrutura de rede.

O modelo de Computação em Nuvem foi desenvolvido com o objetivo de fornecer serviços de fácil acesso e de baixo custo e garantir características tais como disponibilidade e escalabilidade. Com isso, os usuários dos serviços não precisam conhecer aspectos de localização física e de entrega dos resultados dos mesmos.

[1] Datacenter é o local onde são concentrados os equipamentos de processamento e armazenamento de dados de uma empresa ou organização.

[2] Serviço de armazenamento e sincronização de arquivos apresentado pela Google.

Segundo Calheiros (2011), diversos tipos de serviços podem ser oferecidos em ambientes computacionais em nuvem e cada um possui diferentes exigências de composição, configuração e implantação. Assim, a realização de testes em ambientes reais para avaliar o desempenho destes serviços sob condições variáveis, frequentemente, não é uma tarefa viável, visto que demanda muito tempo e é limitada pela rigidez física da estrutura.

Por outro lado há diversas ferramentas que facilitam a reprodução de resultados ao permitirem testar distintos cenários em ambientes repetíveis e controláveis antes da implantação dos serviços em infraestruturas reais. A motivação desse trabalho surge da necessidade de estudar e implementar uma framework[3] Web para testar métodos de alocação de recursos físicos de uma nuvem computacional, mais especificamente a forma de entrada das Máquinas Virtuais em Datacenters na Nuvem.

Sua produção consiste no uso de tecnologias e linguagens de programação, como PHP, JavaScript, HTML e CSS. Estes dão suporte e tornam possível a modelagem das diversas possibilidades de cenários para alocação de recursos físicos. Os resultados obtidos através do uso da ferramenta servem como previsão da situação de um Datacenter após o método de alocação ter sido executado.

[3] Abstração que une códigos comuns entre vários projetos de software provendo uma funcionalidade genérica.

2 OBJETIVOS

2.1 OBJETIVO GERAL

Desenvolver uma ferramenta Web para modelagem de algoritmos de alocação de Máquinas Virtuais nos Servidores de um Datacenter comportados na Nuvem.

2.2 OBJETIVOS ESPECÍFICOS

- Realizar pesquisa bibliográfica dos assuntos pertinentes ao tema.
- Utilizar uma arquitetura de projetos que favoreça o ambiente de desenvolvimento.
- Desenvolver uma interface visual de fácil compreensão para facilitar a visualização dos resultados.
- Realizar testes da ferramenta utilizando um algoritmo descrito pelas literaturas estudadas em uma dimensão de recursos das Máquinas Virtuais.

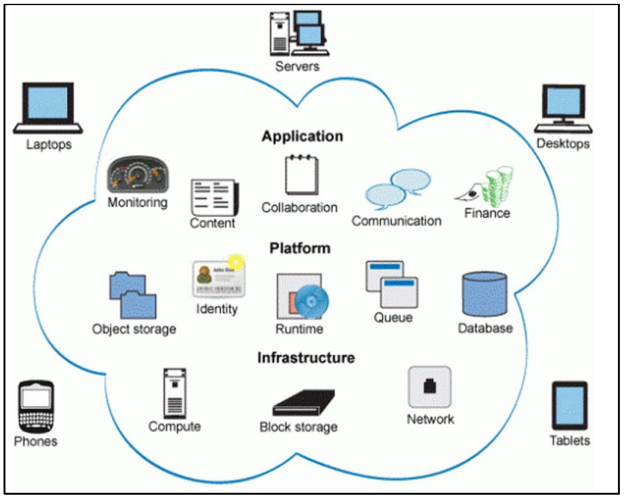
3 REFERENCIAL TEÓRICO

3.1 NUVEM

A Computação em Nuvem está conquistando seu espaço dentro da indústria de TI. A nuvem é um sentido figurado para Internet ou infraestrutura de comunicação entre componentes de uma arquitetura, ocultando sua complexidade. Cada parte de sua infraestrutura é provida em forma de serviço e estes são geralmente alocados em data-centers utilizando hardware compartilhado para processamento e armazenamento.

A Figura 1 mostra a visão geral de uma nuvem computacional. Cada usuário individual se conecta a partir de um computador ou aparelho portátil através da internet. Para esses usuários, a nuvem é visualizada como uma única aplicação, criando assim, a impressão de que o hardware é invisível.

Figura 1 - Visão geral de uma nuvem computacional



Fonte: IMASTERS (2012)

Armbrust (2009) propõe a definição de Computação em Nuvem como um conjunto de serviços de rede que oferece qualidade de serviço sob demanda de forma simples e pervasiva.

Já a definição de NIST (2011, p.2):

> “A computação em nuvem é um modelo para habilitar o acesso por rede ubíquo, conveniente e sob demanda a um conjunto compartilhado de recursos de computação (como redes, servidores, armazenamento, aplicações e serviços) que possam ser rapidamente provisionados e liberados com o mínimo de esforço de gerenciamento ou interação com o provedor de serviços“.

Ainda segundo NIST (2011), uma nuvem deve apresentar os seguintes conceitos essenciais:

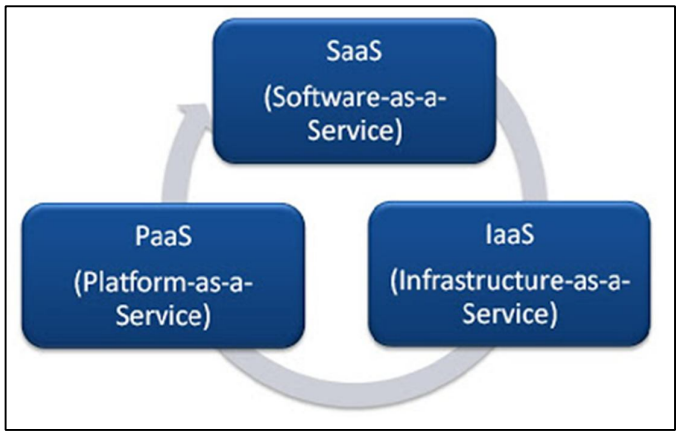
- Autoatendimento sob demanda: funcionalidades computacionais são providas automaticamente sem a interação humana com o provedor de serviço;
- Amplo acesso a serviço de rede: recursos computacionais estão disponíveis através da internet e são acessados via mecanismos padronizados, para que possam ser utilizados por dispositivos móveis e portáteis e computadores;
- Pool de recursos: recursos computacionais (físicos ou virtuais) do provedor são utilizados para servir a múltiplos usuários, sendo alocados e realocados dinamicamente conforme a demanda;
- Elasticidade rápida: as funcionalidades computacionais devem ser rápidas e elasticamente providas, assim como rapidamente liberadas. O usuário dos recursos deve ter a impressão de que ele possui recursos ilimitados, que podem ser adquiridos (comprados) em qualquer quantidade e a qualquer momento. A elasticidade tem três principais componentes: escalabilidade linear, utilização sob demanda e pagamento por unidades consumidas de um recurso;
- Serviços mensuráveis: os sistemas de gerenciamento utilizados pela computação em nuvem controlam e monitoram automaticamente os recursos para cada tipo de serviço (armazenamento, processamento e largura de banda). Esse monitoramento do uso dos recursos deve ser transparente para o provedor de serviços, assim como para o consumidor do serviço utilizado.

Este trabalho utilizará a visão do NIST, que descreve a computação em nuvem composta por além cinco características anteriores, três modelos de serviço e quatro modelos de implantação. Esses modelos serão descritos detalhadamente a seguir.

3.1.1 Modelos de serviço

A computação em nuvem distribui os recursos na forma de serviços. Com isso, podemos dividir a computação em nuvem em três modelos, em relação aos serviços oferecidos. A ilustração a seguir da Figura 2 apresenta sua estrutura, formada por Saas (Software como Serviço), Paas (Plataforma como Serviço) e IaaS (Infraestrutura como Serviço):

Figura 2 - Modelos de serviço de nuvem.

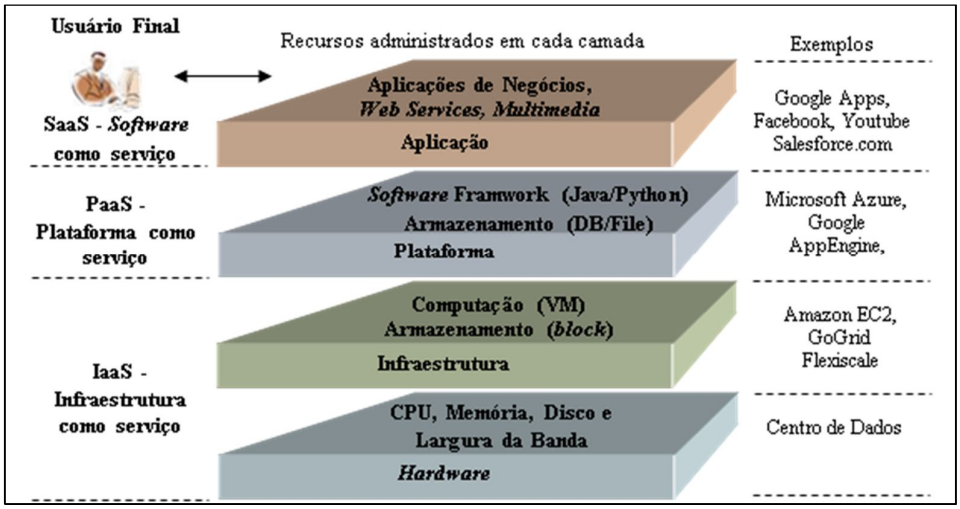


Fonte: Eashani Rodrigo (2011).

As três estruturas serão descritas mais detalhadamente na próxima seção.

A Figura 3 mostra a arquitetura da computação em nuvem que é dividida em quatro camadas, a camada de aplicação, plataformas, infraestrutura e hardware, sendo situados dentro dessas camadas os três modelos de serviços.

Figura 3 - Arquitetura de computação em nuvem.



Fonte: Adaptado de Zhang et al. (2010).

3.1.1.1 *Software como um Serviço (SaaS)*

De acordo com Sousa, Moreira e Machado (2009) o modelo SaaS proporciona softwares acessíveis a partir de vários dispositivos por meio de uma interface através de um navegador Web. Além disso, o usuário não administra ou controla a rede, servidores, sistemas operacionais, armazenamento ou mesmo as características individuais da aplicação, exceto por algumas configurações específicas que são moldadas para seu objetivo. Assim, os desenvolvedores oferecem uma camada de serviço sob demanda a qual seu pagamento é feito conforme o uso.

Como as aplicações estão na Web, podem ser facilmente acessadas por usuários independente do lugar e momento, permitindo uma melhor integração entre as unidades de uma mesma empresa ou outros serviços de software. Dessa forma, novos recursos podem ser acoplados sem que o usuário perceba.

O SaaS diminui os custos pois não é dispensada da aquisição de diversas licenças de software.

Além disso, não exige que o cliente tenha uma estrutura complexa de tecnologia, dispensando investimentos no treinamento de pessoal para manutenção e instalação de software.

Alguns exemplos desse tipo de serviço são:

- Netflix: oferece serviço de TV por Internet a mais de 50 milhões de assinantes distribuídos por mais de 40 países.
- Salesforce: empresa americana de software sob demanda, mais conhecida por ter produzido o CRM (Customer Relationship Manager) com o mesmo nome da empresa.
- Paypal: sistema que permite a transferência de dinheiro entre indivíduos ou negociantes usando um endereço de e-mail.

3.1.1.2 *Infraestrutura como um Serviço (IaaS)*

Semelhante ao anterior, o usuário contrata uma infraestrutura em forma de serviço. Este modelo tem por principal objetivo tornar mais fácil e acessível o fornecimento de recursos, como: servidores, rede, armazenamento e outros recursos de computação fundamentais. Nesses recursos, o usuário pode instalar e executar softwares arbitrários, podendo incluir sistemas operacionais e diversas aplicações.

Em geral, segundo Sousa, Moreira e Machado (2009), o termo IaaS se refere a uma infraestrutura computacional fundamentada em técnicas de virtualização de recursos de computação. Esta infraestrutura pode escalar dinamicamente, aumentando ou diminuindo os recursos de acordo com as necessidades das aplicações. A virtualização será discutida em um dos tópicos posteriores.

Este modelo também é responsável por prover toda a infraestrutura necessária para a SaaS e o PaaS escalando dinamicamente, aumentando ou diminuindo os recursos de acordo com a necessidade das aplicações.

Do ponto de vista de economia e aproveitamento, dispensa a compra de novos servidores e equipamentos de rede para a ampliação de serviços, podendo-se aproveitar os recursos disponíveis e adicionar novos servidores virtuais à infraestrutura existente, utilizando técnicas de virtualização.

Alguns exemplos de serviços associados a este modelo são:

- Amazon EC2 (Elastic Cloud Computing): compõe uma parte central da plataforma de cloud computing da Amazon Web Services (AWS). Permite que os usuários aluguem computadores virtuais nos quais rodam suas próprias aplicações.

- Eucalyptus (Elastic Utility Computing Architecture Linking Your Programs To Useful Systems): implementa nuvens privadas e, por ser um ambiente compatível com o EC2 da Amazon, permite que as nuvens privadas criadas por ele, interajam com a nuvem publica da Amazon, pois usa o mesmo interface de programação.

3.1.1.3 *Produto como um Serviço (PaaS)*

Segundo Sousa, Moreira e Machado (2009) a PaaS oferece uma infraestrutura de alto nível de integração para implementar e testar aplicações na nuvem. Sendo o principal modelo de serviço, o ambiente é fornecido pelo provedor junto às ferramentas de desenvolvimento, disponibilização e controle das aplicações. Assim, ele oferece infraestrutura para testar e implementar aplicações dentro da nuvem.

Do ponto de vista empresarial, ele permite o uso de serviços de terceiros ao passo que aumenta o uso de modelo de suporte no qual os usuários se inscrevem para solicitações de serviços de TI ou de resoluções de problemas pela Web. Dessa forma, descentraliza-se uma grande carga de trabalho e responsabilidade pertencentes à equipe de TI.

São exemplos de PaaS:

- Windows Azure: totalmente hospedado e controlado pela Microsoft, qualquer desenvolvedor cadastrado pode enviar seus aplicativos para o Azure e rodá-los diretamente através do serviço.
- Google App Engine: plataforma de Computação em nuvem para desenvolver e hospedar aplicações web na infraestrutura da Google.
- Aneka: Age como uma ferramenta de desenvolvimento para implementar aplicações customizadas e implantá-las tanto em nuvens públicas como privadas.

3.1.2 Modelos de Implantação

Os modelos de implantação dependem da necessidade das aplicações que serão implementadas. A restrição ou abertura de acesso depende do processo de negócios, do tipo de informação e do nível de visão desejado. Algumas

organizações tem preferência em não deixar que todos os seus usuários tenham acesso a determinados recursos de seu ambiente. Assim, surge a necessidade de ambientes mais restritos. Segundo NIST (2011) são quatro os modelos de implantação: privado, público, comunitário, híbrido. Estes modelos serão abordados de forma mais detalhada nos tópicos a seguir.

3.1.2.1 *Privado*

Neste modelo a infraestrutura da Nuvem é utilizada somente para a organização. Porém, podendo ser geridos pela própria ou por terceiros. Esses tipos de nuvem são exclusivamente para um tipo de usuário. Dessa forma, são empregadas políticas de acesso aos serviços.

As técnicas utilizadas para prover tais características podem ser em nível de gerenciamento de redes, configurações dos provedores de serviços e a utilização de tecnologias de autenticação e autorização.

3.1.2.2 *Público*

A infraestrutura da Nuvem é disponibilizada a qualquer tipo de usuário ou empresa, desde que conheça a localização do serviço. Assim, pode-se inferir que é propriedade de uma organização da venda de serviços em nuvem.

Neste modelo de implantação não são criadas restrições de acesso quanto ao gerenciamento de redes ou técnicas de autenticação e autorização. Porém, essa liberdade pode gerar problemas de segurança devido ao grande público que a acessa.

3.1.2.3 *Comunitário*

Uma Nuvem comunitária compartilha a infraestrutura entre várias organizações de interesses comuns (segurança, jurisdição, conformidade), sejam administrados internamente ou por terceiros e podem ser hospedados internamente ou externamente. Pode ser administrado por organizações ou de um terceiro e pode existir no local ou remota.

Seus custos são distribuídos por menos usuários do que uma nuvem pública (mas mais do que uma nuvem privada), portanto, apenas alguns dos benefícios da computação em nuvem são realizados com efetividade.

3.1.2.4 *Híbrido*

A Nuvem híbrida é uma composição de duas ou mais nuvens que permanecem suas entidades únicas e são unidas oferecendo os benefícios da implantação de modelos múltiplos.

Segundo NIST (2011) elas permitem que uma nuvem privada possa ter seus recursos ampliados a partir de uma reserva de recursos em uma nuvem pública. Essa característica possui a vantagem de manter os níveis de serviço mesmo que haja flutuações rápidas na necessidade dos recursos. A conexão entre as nuvens pública e privada pode ser usada até mesmo em tarefas periódicas que são mais facilmente implementadas nas nuvens públicas, por exemplo.

3.2 VIRTUALIZAÇÃO

O conceito de virtualização surgiu por volta de 1960 através dos mainframes[4]. Esses computadores criavam múltiplos servidores virtuais que compartilhavam hardware, de forma que naquela época os custos de hardware eram demasiadamente caros e somente um pequeno público utilizava-os.

Este conceito, também conhecido como Logical Partition Programming[5], permite a divisão de um único servidor em vários outros ou partições virtuais totalmente independentes entre si. Cada uma dessas partições pode executar sistemas operacionais distintos e dimensionar o hardware dependendo da necessidade.

Em sua essência, de acordo com Carissimi (2009), a virtualização consiste em estender ou substituir um recurso ou uma interface existente por outra de modo a imitar um comportamento.

[4] Computador de grande porte dedicado normalmente ao processamento de um volume grande de informações.

[5] Subconjunto de recursos de hardware do computador, virtualizados como um computador separado.

Em virtude do crescimento dessa técnica, para Agostinho (2009), são resultantes as seguintes características:

- Redução de tempo de indisponibilidade dos servidores e dispensa a necessidade de compra de equipamentos, restringindo somente para o caso de ser realmente necessária a reposição.
- A maioria dos produtos de criação de cópias de segurança/restauração suportam a recuperação dos Sistemas Operacionais e aplicações dos servidores físicos instalados como Máquinas Virtuais (MVs).
- O uso de Virtual Desktops[6] pode reduzir custos, ao mesmo passo que permite manter o controle do ambiente cliente fornecendo camadas adicionais de segurança sem custos adicionais.
- As MVs são ideais para criação e distribuição de ambientes de teste para viabilidades de soluções de forma segura e confiável. Caso esse ambiente criado obtiver sucesso, sua transferência para um ambiente produtivo torna-se fácil e relativamente rápida dispensando a reconstrução do sistema.
- A facilidade de gerenciamento oferecido pelas soluções de virtualização diminui o número de máquinas físicas, facilitando o trabalho dos técnicos responsáveis.
- A utilização eficiente dos recursos de processamento, memória e espaço de armazenamento e físico. Desta forma, criam-se vários sistemas heterogêneos ou não, isolados em um único hardware.

A virtualização permite que através da criação de cópias das máquinas virtuais criadas, além da criação de pontos de restauração seja possível que o próprio monitor de máquina virtual, ao perceber que uma máquina está indisponível, restabeleça essa máquina em outro ambiente virtualizado controlado por outro monitor de máquina virtual automaticamente.

Os tópicos a seguir descreverão os conceitos relacionados a Máquinas Virtuais.

[6] Usado para descrever formas em que o espaço virtual do ambiente de trabalho de um computador é expandido para além dos limites físicos da área de exibição do monitor através da utilização de software.

3.2.1 Máquinas Virtuais

Máquinas virtuais ilustram uma abstração que enxerga desde recursos físicos a Sistemas Operacionais (SOs). Normalmente, nestes sistemas é executado apenas um único SOs sobre uma infraestrutura física. Um dos pontos explorados pelas técnicas de virtualização é que as Máquinas Virtuais incluem uma camada abstrata intermediária permitindo a execução de vários SOs sobre uma mesma plataforma de hardware. Um dos conceitos de máquinas virtuais segundo Popek e Goldberg (1974), é: “Uma máquina virtual é vista como uma duplicata eficiente e isolada de uma máquina real“.

Estas máquinas podem ser caracterizadas em dois tipos:

- Processos: conhecidas também como MVs de aplicação, são construídas para dar suporte de execução a apenas um processo ou aplicação específica.
- Sistemas: construídos para emular uma plataforma de hardware completa, com processador e periféricos, executando múltiplos processos ou aplicações.

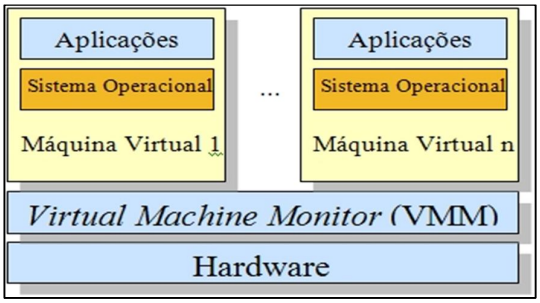
O uso de máquinas virtuais é datado da década de 60, entretanto, apenas há poucos anos atrás o uso da virtualização ganhou proporções maiores. Dessa forma, surgiram novos conceitos e técnicas. Um desses conceitos é o Monitor de Máquinas Virtuais (MMV).

3.2.2 Monitor de Máquinas Virtuais

O MMV tem como função principal monitorar e controlar as máquinas virtuais de um ambiente. Essas MVs podem conter diferentes Sistemas Operacionais executados sobre uma mesma infraestrutura física. Um MMV é responsável por intermediar os acessos aos recursos do sistema.

Uma Máquina Virtual é uma instância do sistema sob controle do MMV. A Figura 4 mostra as camadas desde a infraestrutura física até as aplicações e, além disso, mostra também um MMV controlando “n” instâncias de um SO e suas aplicações, portanto, “n” Máquinas Virtuais.

Figura 4 - Relacionamento das Máquinas Virtuais e do MMV.



Fonte: Diogo Menezes Ferrazani Mattos (2008)

O objetivo do MMV, através da gerência do acesso aos recursos de hardware disponíveis, é compartilhá-los de maneira que vários Sistemas Operacionais sejam executados simultaneamente em um mesmo ambiente.

Segundo Menascé (2005) as vantagens do uso de MVs, são:

- Segurança: Usando máquinas virtuais, pode ser definido qual é o melhor ambiente para executar cada serviço, com diferentes requerimentos de segurança, ferramentas diferentes e o sistema operacional mais adequado para cada serviço. Além disso, cada máquina virtual é isolada das demais. Usando uma máquina virtual para cada serviço, a vulnerabilidade de um serviço não prejudica os demais.
- Confiança e disponibilidade: A falha de um software não prejudica os demais serviços.
- Custo: A redução de custos é possível de ser alcançada com a consolidação de pequenos servidores em outros mais poderosos. Essa redução pode variar de 29% a 64%.
- Adaptação às diferentes cargas de trabalho: Variações na carga de trabalho podem ser tratadas facilmente. Ferramentas autônomas podem realocar recursos de uma máquina virtual para a outra.
- Balanceamento de carga: Toda a máquina virtual está encapsulada no MMV. Sendo assim é fácil trocar a máquina virtual de plataforma, a fim de aumentar o seu desempenho.
- Suporte a aplicações legadas: Quando uma empresa decide migrar para um novo Sistema Operacional, é possível manter o sistema operacional antigo sendo executado em uma máquina virtual, o que reduz os custos com a migração. Vale ainda lembrar que a virtualização pode ser útil para aplicações

que são executadas em hardware legado, que está sujeito a falhas e tem altos custos de manutenção. Com a virtualização desse hardware, é possível executar essas aplicações em hardwares mais novos, com custo de manutenção mais baixo e maior confiabilidade.

3.3 GESTORES DE NUVEM

Os gestores de nuvem têm como papel principal coordenar e monitorar os Monitores de Máquinas Virtuais (também conhecidos como Hypervisors) implantados na estrutura física. As formas de interação com a nuvem permitem ao usuário instanciar e gerenciar MVs sob demanda. Alguns dos mais importantes e conhecidos gestores serão descritos nos tópicos a seguir.

3.3.1 Amazon AWS

O Amazon AWS é um ambiente de computação em nuvem com características de escalabilidade, disponibilidade, elasticidade e desempenho. Segundo Zhang (2010), ela provê um conjunto de serviços em nuvem de forma que seus usuários possam implantar aplicativos e serviços sob demanda.

O Amazon Web Services é composto por um conjunto de sistemas, dentre os quais podemos destacar:

- Execução: Elastic Compute Cloud (EC2): sistema responsável pelo gerenciamento da execução de aplicações na infraestrutura da Amazon, permitindo um controle completo das instâncias dos sistemas, sendo possível acessar e interagir com cada umas destas, de forma similar a máquinas convencionais. Também é possível escolher as características de cada instância, tais como sistema operacional, pacotes de software e as configurações das máquinas, como CPU, memória e armazenamento. Para garantir a segurança, utiliza firewall para controlar o acesso às instâncias, criando ambientes virtuais privados.

- Armazenamento: Simple Storage Service (S3), SimpleDB e Relational Database Service (RDS): sistema de arquivos distribuído, utilizado para recuperar e armazenar dados. Durante a execução, as tarefas recuperam os arquivos no S3 e realizam o devido processamento. As soluções EC2 armazenam arquivos como objetos no S3 e todos os metadados relacionados ao objeto no SimpleDB. O SimpleDB fornece as funcionalidades de um sistema banco de dados como armazenamento, indexação e consultas em ambientes de nuvem.
- Programação: Simple Queue Service (SQS) e Elastic MapReduce: fornece automação de workflows trabalhando em conjunto com o EC2. O SQS enfileira e armazena mensagens que estão sendo trocadas entre os recursos computacionais. Os desenvolvedores podem, de maneira simples, mover dados entre os recursos distribuídos das aplicações de diferentes domínios com garantia de entrega.
- Monitorização: Cloudfront: enfoque na entrega conteúdo estático (imagens, vídeos, arquivos JavaScript, etc) com baixa latência.

3.3.2 Eucalyptus

Segundo Liu et al. (2007), o Eucalyptus é uma infraestrutura de código aberto que fornece uma interface compatível com o Amazon EC2, S3, Elastic Block Store (EBS) e permite aos usuários criarem uma infraestrutura e experimentar a computação em nuvem.

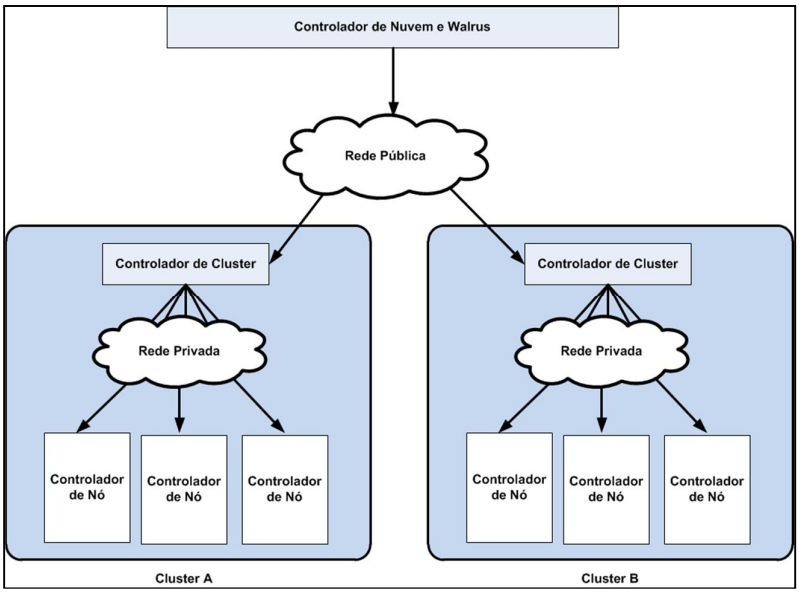
Segundo Nurmi et al. (2009), sua arquitetura é divida em quatro partes:

- Controlador de nó: Controla instâncias das máquinas virtuais nos nós.
- Controlador de Cluster: Ponte de comunicação entre controlador de nó e o controlador de nuvem.
- Controlador de armazenamento (Walrus): Gerencia tráfego de dados dentro e fora da nuvem.
- Controlador de nuvem: Controla a nuvem como um todo.

A arquitetura do Eucalyptus é modular e, como possui código aberto, facilita que pesquisadores desenvolvam diversas modificações à sua estrutura original. A

Figura 5 mostra a sua estrutura e seus componentes. Esses componentes são implementados como serviços web.

Figura 5 – Arquitetura do Ecalyptus.



Fonte: Adaptada de (Nurmi et al., 2009)

O Eucalyptus tem como objetivo auxiliar a pesquisa e o desenvolvimento de tecnologias para computação em nuvem e possui as seguintes características:

- Interface compatível com o EC2, instalação e implantação simples usando ferramentas de gerenciamento de clusters,
- Conjunto de políticas de alocação extensível de nuvem, sobreposição de funcionalidade que não requer nenhuma modificação em ambiente Linux,
- Ferramentas para administrar e auxiliar a gestão do sistema e dos usuários e capacidade de configurar vários clusters, cada um com endereços privados de rede interna em uma única nuvem.

Esse sistema permite aos usuários iniciar, controlar o acesso e gerenciar todas as máquinas virtuais utilizando uma emulação do Protocolo Simples de Acesso a Objeto (SOAP) do Amazon EC2 e interfaces de consulta. Neste sentido, os usuários interagem com o Eucalyptus utilizando as ferramentas e interfaces exatamente do mesmo modo que eles interagiriam com o Amazon EC2.

3.3.3 Microsoft Azure

Segundo Sousa (2009) o Microsoft Azure é uma plataforma para execução de aplicativos e serviços, baseada nos conceitos da computação em nuvem. A plataforma Azure é formada pelo sistema operacional Windows Azure e um conjunto de serviços: Live Services, .NET Services, SQL Services, SharePoint Services e Dynamics CRM Services.

Estes serviços são definidos da seguinte forma:

- Microsoft .NET Services: conjunto de serviços escaláveis, orientados ao desenvolvedor e que oferecem os componentes necessários para a maioria das aplicações baseadas em nuvem.
- Live Services: conjunto de componentes dentro do Azure para o tratamento de dados do usuário e recursos da aplicação. Live Services possibilita aos desenvolvedores construírem aplicações ricas que podem conectar com usuários do Windows Live. O Live Services inclui as tecnologias do Live Mesh para sincronização de dados dos usuários e possibilita a extensão de aplicações Web entre múltiplos dispositivos.
- SQL Services: serviço de armazenamento de dados e de processamento de consultas escalável, sendo construído com base na tecnologia do SQL Server.
- SharePoint Services: permite colaborar e criar aplicações Intranet e o Dynamics CRM Services é um sistema totalmente integrado de CRM.

O Windows Azure é um sistema operacional para serviços na nuvem que é utilizado para o desenvolvimento, hospedagem e gerenciamento dos serviços dentro do ambiente Azure. Cada aplicação desenvolvida no Windows Azure é executada em uma ou mais máquinas virtuais. O objetivo principal desta tecnologia é prover escalabilidade massiva e para isso o Windows Azure cria diferentes instâncias iguais que executam o mesmo tipo de tarefas.

3.4 ALOCAÇÃO DE MÁQUINAS VIRTUAIS EM DATACENTERS

Pode-se inferir que uma Máquina Virtual está alocada quando consome recursos físicos de um servidor, este, pertencente a um datacenter. Em um ambiente

de computação nas nuvens, os usuários enviam requisições ao centro de dados para alocação de recursos em seus servidores. Para atender a demanda dos clientes, o centro de dados aloca máquinas virtuais nos servidores físicos de sua infraestrutura. Dessa forma, o problema se encontra em determinar qual servidor receberá a MV.

Segundo Hyser (2007), deve-se conhecer a demanda de recursos de cada MV, o ambiente de execução e as políticas específicas do Datacenter para possíveis conflitos.

Para Mishra e Sahoo (2011) a alocação de uma MV é dividida em duas etapas: inicialmente é estimada a demanda de recursos da máquina virtual; em seguida, escolhe-se o servidor onde essa máquina virtual será alocada. Porém, há certa dificuldade em estimar os recursos exigidos por uma MV, pois estes podem variar durante sua execução.

Hyser (2007) diz que as métricas devem levar em consideração as várias dimensões das MVs como as capacidades do processador, memória, disco rígido e banda de rede, além disso, para Buyya (2010), deve-se incluir o consumo de energia nesta lista.

3.4.1 Paradigmas e algoritmos de Alocação

Segundo Singh (2008) a alocação de Máquinas Virtuais em Datacenters é um problema do tipo NP-difícil. Um dos problemas nos mecanismos de alocação está no fato da complexidade temporal para alocação de um conjunto de n ($n \in \{1,\cdots,n\}$) máquinas virtuais em um conjunto de m servidores ($m \in \{1,\cdots,m\}$). Uma solução intuitiva conduz a utilização de um algoritmo guloso, onde, para cada máquina virtual, percorrem-se todos os servidores para selecionar em qual deles deve ser feita a alocação. Este tipo de algoritmo tem complexidade temporal da ordem O(n x m).

Segundo Buyya, Beloglazov e Abawaji (2012) para que seja determinada a escolha do Servidor {m} a ser alocada a MV {n}, deve-se seguir os seguintes critérios:

- Servidor com menor quantidade de recursos não utilizados: deve-se analisar o Servidor que possui maior quantidade de recursos livres e alocar neste servidor, desde que seja suficiente à MV.

- Servidor com maior quantidade de recursos não utilizados: segue a mesma ideia do anterior, porém, atentando-se ao Servidor com maior quantidade de recursos não utilizados de modo a alocar as MVs o mais disperso possível.
- Aleatório: seleciona-se um servidor aleatoriamente e aloca a máquina virtual, isso se o servidor possuir recursos suficientes para receber a máquina virtual.
- Menor consumo de energia: com esse critério, busca-se o servidor onde o impacto energético da alocação da máquina virtual seja o menor.

Estes critérios são fator essencial para o entendimento dos algoritmos existentes de alocação. Segundo Seiden (2002), os algoritmos mais comuns para realizar a alocação de Máquinas Virtuais em Datacenters, são:

- First-Fit: a MV será alocada no primeiro Servidor com capacidade disponível.
- Next-Fit: A MV será alocada no próximo Servidor com capacidade disponível.
- Best-Fit: A MV será alocada no recipiente que resultar na menor capacidade residual, ou seja, aquela que resultar em um melhor preenchimento do Servidor.
- Worst-Fit: A MV será alocada no recipiente que resultar na maior capacidade residual, ou seja, naquele que resultar na maior capacidade disponível após a alocação.

Os autores Buyya, Beloglazov e Abawaji (2012), citam os seguintes critérios de alocação:

- Servidor com menor quantidade de recursos não utilizados: analisa-se qual servidor possui a menor quantidade de recursos não utilizados e realiza-se a alocação, desde que esses recursos sejam suficientes para a alocação da máquina virtual. Nesse caso, busca-se a consolidação das máquinas virtuais na menor quantidade de servidores possível.
- Servidor com maior quantidade de recursos não utilizados: mesmo procedimento do item anterior, diferenciando-se apenas pela escolha do servidor que possui a maior quantidade de recursos não utilizados. Desse modo, tenta-se alocar as máquinas virtuais do modo mais disperso possível.
- Aleatório: seleciona-se um servidor aleatoriamente e aloca a máquina virtual, isso se o servidor possuir recursos suficientes para receber a máquina virtual.
- Menor consumo de energia: com esse critério, busca-se o servidor onde o impacto da alocação da máquina virtual seja o menor.

3.5 TECNOLOGIAS WEB

No início da Internet as páginas eram compostas basicamente de páginas estáticas, ou seja, páginas que dia após dia tinham o mesmo conteúdo, e que não interagiam com seus visitantes.

Essas linguagens eram limitadas quanto ao seu uso e eram pouco amigáveis. Mas estas iniciativas permitiram que outras linguagens fossem desenvolvidas exclusivamente para Internet. Entre elas, pode-se citar: PHP, Java, Javascript e outras, sendo que algumas destas linguagens rodam no lado Servidor (Server-Based ou Back-End) e outras rodam no lado Cliente (Client-Based ou Front-End).

3.5.1 Linguagens de Programação

Linguagens de Programação são linguagens utilizadas no desenvolvimento de interfaces ou aplicativos para a Internet, podendo rodar em um servidor, ou diretamente do computador do cliente. Os tópicos a seguir descrevem de forma mais ampla algumas dessas linguagens.

3.5.1.1 *HTML e CSS*

Segundo Ferreira e Eis (2010), o HTML (Hyper Text Mark-up Language) é baseado no conceito de Hipertexto. Hipertexto são conjuntos de elementos (ou nós ligados por conexões). Estes elementos podem ser palavras, imagens, vídeos, áudio, documentos etc. Estes elementos conectados formam uma grande rede de informação. Eles não estão conectados linearmente como se fossem textos de um livro, onde um assunto é ligado ao outro seguidamente. A conexão feita em um hipertexto é algo imprevisto que permite a comunicação de dados, organizando conhecimentos e guardando informações relacionadas.

Ainda segundo os autores ele foi criado para ser uma linguagem independente de plataformas, browsers e outros meios de acesso.

O documento criado pelo HTML é formatado pelo CSS (Cascading Style Sheets). As folhas de estilo CSS são definidas pelo W3C (1999) como “um mecanismo simples para acrescentar estilo a documentos Web”, isto é, CSS é uma

linguagem de layout padrão para a Web que controla cores, tipografia, tamanhos e posicionamento de imagens.

3.5.1.2 *PHP*

O Manual do PHP (2003) descreve o PHP (Hypertext Preprocessor) como uma linguagem de programação de ampla utilização, interpretada, que é especialmente interessante para desenvolvimento para a Web e pode ser mesclada dentro do código HTML. O PHP pode ser utilizado na maioria dos Sistemas Operacionais (Linux, Windows, Mac OSX) e também nos servidores Web atuais como o Apache, Microsoft Internet Information Server e outros.

Segundo Buyens (2000), o PHP é uma linguagem livre, licenciada conforme a Licença Pública GNU (GPL), ou seja, todo usuário possui a liberdade de executar, copiar, distribuir, estudar, modificar e aperfeiçoar seu software.

Em sua versão 5, segundo Zandstra (2006), representa, explicitamente, o endosso aos objetos e à Programação Orientada a Objetos, apesar de ser contextualizada como linguagem procedural.

A cada dia que passa, o PHP se torna a solução de empresas, entre outras, pode-se citar algumas das razões segundo Kabir (2002):

- Desenvolvimento Web rápido;
- Portável nas principais plataformas de sistemas operacionais;
- Portável nas principais plataformas de servidores Web;
- Suporte interno para os Bancos de Dados mais importantes;
- Alto desempenho;
- Fácil aprendizado para novos desenvolvedores;
- Suporte para padrões da Internet;
- Amigável para as empresas;
- Comunidade de fonte aberta.

3.5.1.3 *JavaScript*

Javascript é uma linguagem “que liga uma página web a uma experiência interativa.” (Morrison, p.4, 2008) proporcionando ao cliente uma interatividade maior

com os recursos que uma página web proporciona. Códigos JavaScript rodam no browser, ou seja, rodam no lado do cliente. A tecnologia JavaScript funciona juntamente com HTML como um dos modernos componentes para a construção de uma página web e entra em ação sempre que solicitado. Suas principais características são:

- As variáveis oferecem tipos dinâmicos, ou seja, não precisam ter seus tipos declarados.
- É uma linguagem interpretada, portanto, o seu código-fonte não precisa ser compilado.
- As funções desenvolvidas utilizando a linguagem JavaScript são executadas no browser do cliente.
- É orientada a eventos, ou seja, as funções são executadas quando um evento associado à função ocorre.
- Possui diversas bibliotecas para auxílio de código.

3.5.1.3.1 *Jquery*

jQuery é uma biblioteca JavaScript que tem o intuito de facilitar a implementação de código. Tem como funcionalidades, seleção de elementos HTML, manipulação de elementos HTML, eventos HTML, efeitos e animações JavaScript. Com a utilização do jQuery, podem-se solucionar problemas de incompatibilidade entre os navegadores, redução do código, e reutilização de códigos já criados por outros desenvolvedores.

3.5.1.3.2 *Ajax*

Segundo Soares (2006, p.17):

> "AJAX é um acrônimo consagrado muito recentemente por Jesse James Garret, da Adaptive Path, e significa Asynchronous JavaScript and XML (JavaScript e XML assíncrono), porém o que temos é muito mais mais que a junção de JavaScript com XML, é todo um conceito de

> navegação e atualização de páginas Web. Algumas partes descritas na definição de AJAX não são novas, as quais muitas vezes foram denominadas de DHTML (HTML Dinâmico) e Script Remoto.”

O AJAX surgiu para resolver um problema que ocorre desde o surgimento da Internet, que é o de que a interação é feita de forma síncrona, ou seja, exige-se que, para cada solicitação em uma página Web, atualize-se a página inteira no navegador. Não importava se a atualização era só de uma pequena parte da página, toda a página era recebida pelo servidor e a página era toda redesenhada e retornada para o navegador Web.

Uma de suas qualidades é o fato de poder trafegar apenas os dados que realmente foram atualizados em uma página Web e assim ganhar em desempenho e interatividade com o usuário.

3.5.2 Padrão de Projeto MVC

Para Zandstra (2006), um padrão de projeto é um problema analisado e uma boa prática para sua solução explicada. Problemas que se repetem são comuns, e padrões de projeto descrevem e formalizam esses problemas e suas soluções, tornando a experiência, obtida a duras penas, disponível para uma comunidade maior de programação.

Segundo Gonçalves (2007, p.141):

> “O MVC é um conceito (paradigma) de desenvolvimento e design que tenta separar uma aplicação em três partes distintas. Uma parte, a Model (Modelo), está relacionada ao trabalho atual que a aplicação administrativa, outra parte, a View (Visão), está relacionada a exibir os dados ou informações dessa uma aplicação e a terceira parte, Controller (Controlador), em coordenar os dois anteriores exibindo a interface correta ou executando algum trabalho que a aplicação precisa completar”.

Para Fox (2006) a Visão é a camada responsável pela apresentação. Ela recebe o estado do Modelo através do Controlador. Os objetos dessa camada recebem os dados de entrada do usuário que serão repassados para o controlador.

Segundo Basham (2008) o Modelo é a camada que abriga a verdadeira lógica de negócio. Possui as regras para obtenção e atualização do estado e fornece ao Controlador o acesso.

O padrão de projetos MVC divide a aplicação em três camadas: Controle, Visão e Modelo. Essas camadas possuem as seguintes responsabilidades:

- Controle: é definida pelo objeto conhecido como Controller ou Controlador e atua sobre os dados apresentados pelo modelo, decidindo como o modelo devera ser alterado ou deve ser revisto e qual apresentação deve ser exibida.
- Modelo: é o objeto que representa os dados do programa. Maneja esses dados e controlam todas suas transformações. Esse modelo não tem conhecimento especifico do Controlador e das Visões, nem sequer contém referência a eles.
- Visão: maneja a apresentação visual dos dados apresentados pelo Modelo e passados pelo Controlador.

4 METODOLOGIA

4.1 REQUISITOS INICIAIS

A implantação do Projeto consistiu na instalação de recursos essenciais para o desenvolvimento da ferramenta. Por ser uma ferramenta Web, foi necessário um ambiente propício ao desenvolvimento nesta plataforma.

Para tanto, foi instalado em um computador convencional com Sistema Operacional Windows 8.1, o navegador Google Chrome (versão 46), necessário para interpretação dos códigos elaborados das linguagens de programação utilizadas no desenvolvimento.

Além disso, foi necessária também, uma ferramenta para edição de texto para facilitar na modelagem de código das linguagens de programação. O programa escolhido foi Sublime Text (versão 3) por não necessitar de configurações adicionais, estando adequado e pronto para uso.

A terceira ferramenta instalada foi o WAMP (versão 2.5). Esta plataforma cria um ambiente de desenvolvimento instalando consigo o Apache Server, necessário para validação deste projeto. Foi criada também uma pasta de trabalho chamada “Alocador”. Esta pasta deve ficar no diretório “/var/www/” do WAMPServer para que seja executado no Servidor local criado pelo software.

Dentro da pasta “Alocador” criada anteriormente, foram criadas as seguintes pastas de trabalho:

- “css”: contém os arquivos de estilos criados.
- “js”: contém as bibliotecas JavaScript utilizadas.
- “controlador”: contém os arquivos da camada de Controle.
- “visão”: contém os arquivos da camada de Visão.
- “modelo”: contém os arquivos da camada de Modelo.

Por último, foi efetuado o download do jQuery (versão 2.1.1). Esta biblioteca deve ser inserida na pasta “js”.

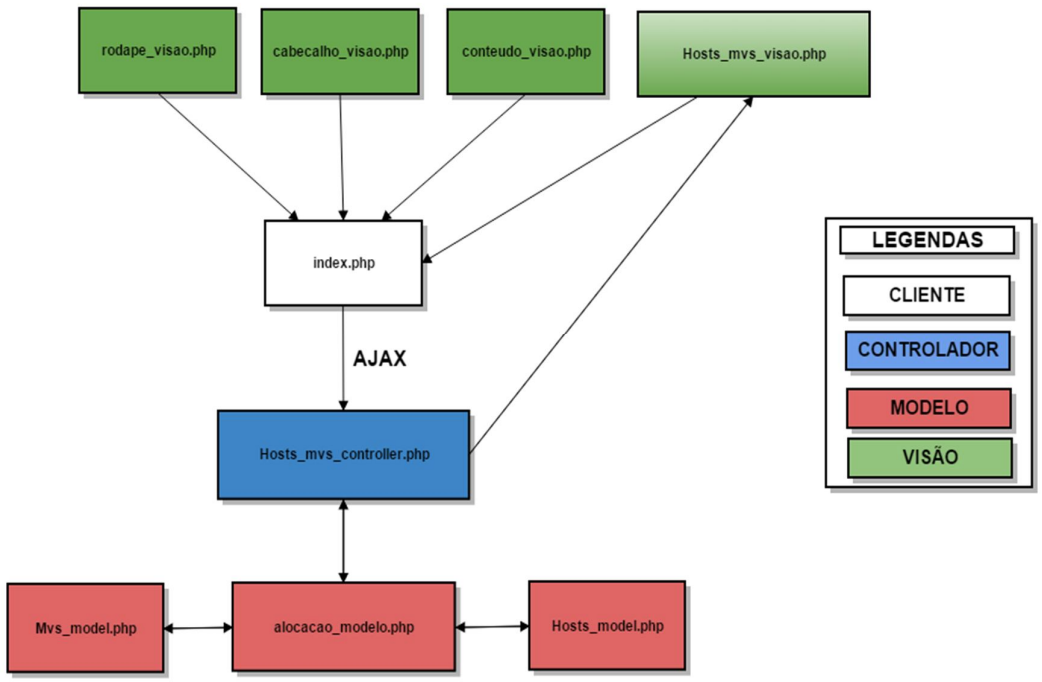
Após a adequação do ambiente de desenvolvimento, foi iniciado o seu processo que será descrito nos tópicos a seguir.

4.2 PROJETO

4.2.1 Arquitetura do Projeto

A parte inicial do desenvolvimento consistiu no planejamento da arquitetura do projeto. Primeiramente, foi escolhido um padrão de projeto baseado no objetivo central: produzir uma ferramenta capaz de oferecer suporte a criação de algoritmos para alocação de MVs em Servidores. Para isso, foi utilizado o padrão de projeto MVC. Este padrão consiste na criação das classes de Controle, Modelo e Visão.
A figura 6 representa um diagrama criado para reproduzir os conceitos da arquitetura MVC deste projeto.

Figura 6 – Diagrama do Projeto



Fonte: Criado pelo Autor (2015)

Conforme demonstra a legenda, na Figura 6 estão representadas as três camadas do padrão MVC e os arquivos que foram criados relacionados a elas. Os tópicos a seguir abordarão de forma mais detalhada a identidade de cada uma dessas camadas.

4.2.1.1 *Camada de Visualização*

A camada de Visão ou Visualização inclui os elementos de exibição do conteúdo HTML, ou seja, a interface do usuário. Sua responsabilidade é receber os dados de entrada e imprimir os de saída.

Os arquivos “rodape_visao.php”, “cabecalho.php” e “conteudo.php” são responsáveis pelos parâmetros de entrada. Estas visões são incluídas no arquivo “index.php” que gera a visualização no cliente (Navegador). Já o arquivo “hosts_mvs_visao.php” é responsável por imprimir a resposta no cliente.

4.2.1.2 *Camada de Controle*

A camada de Controle determina o fluxo da apresentação servindo como uma camada intermediária entre a camada de apresentação e a de lógica. Sua responsabilidade principal é mapear e controlar as ações.

Representada diretamente pelo arquivo “hosts_mvs_controlador.php”, ela cria o diálogo entre o conteúdo que o cliente (navegador) visualiza e as informações geradas pela camada de Modelo.

4.2.1.3 *Camada de Modelo*

A camada de Modelo é a camada de lógica do projeto, para tanto, ela é responsável por:

- Armazenamento, manipulação e geração de dados: os arquivos “hosts_modelo.php” e “mvs_modelo.php” guardarão todas as estruturas de dados que compõe as Máquinas Virtuais e os Servidores do Datacenter.
- Regras de negócio: o arquivo “alocacao_modelo.php” tem como responsabilidade a manipulação destas estruturas de dados dos arquivos de MVs e Hosts. Nele ficam inseridos os algoritmos de alocação de MVs em Hosts e funções que auxiliem todo o processo envolvido.

4.3 IMPLEMENTAÇÃO

Finalizada a definição do padrão de projeto, o próximo passo foi centralizar o uso das linguagens de programação envolvidas na implementação. São descritas as linguagens:

- HTML: Utilizado para construção do esqueleto das páginas Web. Os arquivos da camada de Visão são puramente codificados em HTML, pois devem ser interpretados pelo cliente. Além disso, pequenas alterações serão feitas pelo usuário da ferramenta para que seja implementado um novo algoritmo de alocação.
- CSS: Definição das folhas de estilo do HTML. O seu principal benefício é a separação entre o formato e o conteúdo de um documento. Foi criado um arquivo chamado “estilos.css” que contém todas as definições de estilo (lista de regras) criadas.
- JavaScript: Foi implementada para que os seus scripts pudessem ser executados do lado do cliente e interagissem com o usuário sem a necessidade deste script passar pelo servidor, controlando o navegador de forma simples. Além disso, realiza a comunicação assíncrona (através do Ajax) e alterando o conteúdo do documento exibido sem que a página seja atualizada. Este projeto possui uso da biblioteca jQuery (arquivo jQuery.js), desenvolvida para simplificar a codificação em JavaScript.
- PHP: Seu uso se encontra na implementação do Back-end, mais precisamente das classes de Controle e Modelo. Representa todo o conteúdo lógico utilizando os conceitos de orientação a objetos. Os algoritmos produzidos para a alocação serão codificados nesta Linguagem, portanto, para os usuários será necessário conhecimento prévio a nível intermediário nesta Linguagem de Programação.

4.4 TESTES

Durante a implementação foram executados Testes Unitários para verificar se a especificação caminhou de forma correta. Segundo Bartié (2002), o objetivo deste teste é percorrer o código de cada classe do sistema e verificar os desvios

condicionais, caminhos alternativos de execução, e se ele responde conforme seus requisitos. Além disso, os testes unitários devem avaliar os requisitos funcionais, de usabilidade e de sistema relacionado ao componente.

Foi adotado então o conceito de desenvolvimento orientado por testes. Para isto, todos os métodos e classes foram testados. Segundo Massol e Husted (2005), a implementação dos Testes de Unidade permite que o teste seja executado para provar que a implementação funciona. Em caso de falhas, a implementação foi refeita até que estivesse funcionando conforme especificado.

4.5 VALIDAÇÃO

De forma a validar o uso da ferramenta criada foi implementado um dos algoritmos para alocação citados pela literatura. O algoritmo escolhido foi o método First Fit. Este método aloca Máquinas Virtuais no primeiro Servidor que estiver disponível da lista de Servidores do Datacenter.

Apesar de ser um método simples, através dele é possível coletar diversos cenários de resposta. Com isso, três cenários foram coletados:

- Um Servidor sem nenhuma MV alocada.
- Uma MV que não está alocada em nenhum Servidor.
- Todos os Servidores possuem MVs Alocadas.

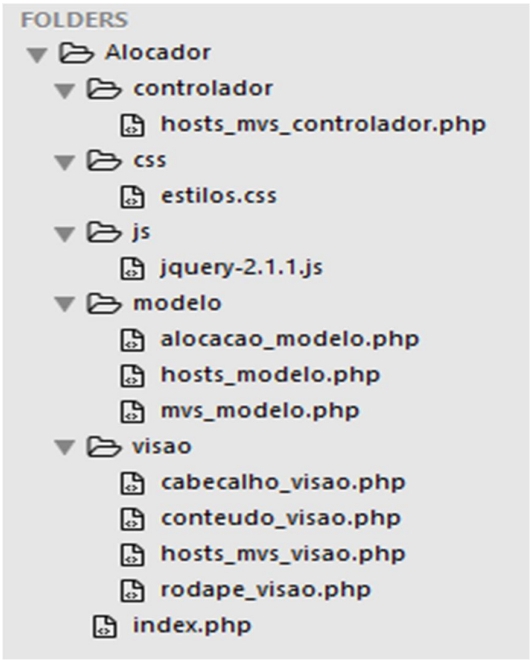
  Isto garante que o método funciona, e, portanto, a ferramenta.

5 RESULTADOS

5.1 AMBIENTE DE TRABALHO

Para que fosse seguido o padrão de projetos MVC, foram inseridos os devidos arquivos demonstrados na Figura 6 em seus devidos diretórios, conforme demonstrado na Figura 7 através do Editor de Texto Sublime Text. Além disso, foram inseridos também os arquivos “estilos.css” e a biblioteca jQuery.

Figura 7 – Diretório de pastas e arquivos no Sublime Text 3.



Fonte: Criado pelo Autor (2015).

5.2 PROJETO E IMPLEMENTAÇÃO

Criados o ambiente de trabalho e seus arquivos, o próximo passo foi de implementar os códigos em suas devidas linguagens de programação. Foram implementados na seguinte ordem: a camada de Visão, a camada de Modelo e por fim, a camada de Controle.

5.2.1 Camada de Visão - Requisição

Para gerar a parte de requisições da camada de Visão (correspondente a Visualização do Usuário) foram necessárias implementações em três documentos da pasta “Visao”. Durante as modelagens foram criados seus estilos no documento de estilos padrão do Projeto encontrado no diretório “/css/estilos.css” do ambiente.

O primeiro documento a ser implementado desta camada é o “cabecalho_visao.php”. Este documento possui o conteúdo do topo da estrutura HTML da visualização. Ele contém o título da ferramenta, as importações de estilos e scripts utilizados no projeto.

Um dos scripts controla a forma que são vistos os formulários, os formulários dos Servidores e MVs somente serão vistos após o preenchimento do formulário inicial. Outro script realiza a requisição Ajax enviando os dados preenchidos nos formulários para a camada de Controle sem que a página seja atualizada.

O segundo documento é o “conteudo_visao.php”. Foram implementados todo o conteúdo HTML da ferramenta Web, de forma mais precisa, os formulários que devem ser preenchidos pelo Usuário de forma a gerar a resposta desejada. Os formulários foram divididos em três visualizações:

- Datacenter: possui os valores de entrada contendo a quantidade de Servidores e a quantidade de Máquinas Virtuais a serem criadas e tipo (método) de alocação desejada. A figura 8 representa este formulário.

Figura 8 – Formulário inicial do Usuário

Fonte: Criado pelo Autor (2015)

- Servidores: possui os valores de entrada das dimensões dos Servidores, ou seja, as quantidades de memória RAM, banda, espaço físico e MIPS (Milhões de Instruções por Segundo). A Figura 9 representa este formulário.

Figura 9 – Formulário de dimensões do(s) Servidor(es)

Fonte: Criado pelo Autor (2015)

- Máquinas Virtuais: possui os valores de entrada das dimensões das MVs. Contém os mesmos campos dos Servidores acrescido do campo “Porcentagem de Carga”, recurso necessário que especifica o uso em % da dimensão de MIPS. A Figura 10 representa este formulário.

Figura 10 – Formulário de dimensões da(s) MV(s)

Fonte: Criado pelo Autor(2015)

O terceiro documento é o “rodape_visao.php” e contém a seção de resposta desejada gerada pelas outras camadas. A resposta final será injetada em uma das tags (rótulos).

Modificados estes arquivos, foram feitas alterações no arquivo “index.php” que deverá conter as importações para os arquivos anteriores. A Figura 11 demonstra como foram feitas as importações através da linguagem PHP.

Figura 11 – Conteúdo do arquivo “index.php”

```php
<?php

include 'visao/cabecalho_visao.php';

include 'visao/conteudo_visao.php';

include 'visao/rodape_visao.php';

?>
```

Fonte: Criado pelo Autor (2015)

5.2.2 Camada de Modelo

A camada de modelo ficou descrita no diretório “/modelo/” e possui três arquivos: “hosts_modelo.php”, “mvs_modelo.php” e “alocacao_modelo.php”. Todos os documentos foram codificados na linguagem PHP, pois representam o Backend da aplicação.

Os dois primeiros arquivos correspondem a estrutura de dados pertencente aos Servidores e as Máquinas Virtuais. Além disso, possuem todos os métodos de get e set que farão o acesso as variáveis.

O arquivo “hosts_modelo.php” possui as seguintes variáveis referentes as características do Servidor:

- ID: número de identificação.
- Ram_servidor: quantidade de memória RAM.
- Banda_servidor: quantidade de banda disponível.
- Espaço_servidor: quantidade de espaço no disco rígido.
- Mips_servidor: quantidade de MIPS.
- Mips_livre_servidor: quantidade de MIPS livre disponível.
- Vms_alocadas_servidor: lista de Máquinas Virtuais alocadas.

Já no arquivo “mvs_modelo.php” estão declaradas as seguintes variáveis das características das Máquinas Virtuais:

- ID: numero de identificação.
- Ram_mv: quantidade de memória RAM alocada.
- Banda_mv: quantidade de banda alocada.
- Espaço_mv: quantidade de espaço alocado.
- Mips_mv: quantidade de MIPS alocada.
- Carga_mv: porcentagem de carga utilizada.
- Mips_real_mv: quantidade de MIPS total após aplicação de porcentagem de carga.
- Servidor_mv: ID do Servidor que está alocada.

O último arquivo, “alocacao_modelo.php”, conta com as regras de negócio desenvolvidas, ou seja, as características relacionadas a alocação destas MVs nos Servidores. Esteve arquivo possui como variáveis a lista de objetos de Hosts e lista de objetos de MVs, além disso, foram criados os seguintes métodos (funções na linguagem PHP):

- verificarRecursosIniciais: esta função verificará se a soma de cada dimensão das MVs a serem criadas é menor do que a soma da dimensão equivalente no Servidor, por exemplo, a soma de MIPS das MVs deve ser sempre menor ou igual a soma de MIPS dos Servidores. Tem como retorno um valor booleano falso ou verdadeiro.
- criarHost: possui parâmetros de entrada das dimensões de um Host (id, ram, banda, espaço e mips), chama os métodos Set criados no arquivo de modelo de hosts e insere na lista de hosts.
- criarMV: possui parâmetros de entrada das dimensões de uma MV (id, ram, banda, espaço, mips, carga), chama os métodos Set criados no arquivo de modelo de MVs e insere na lista de MVs.
- alocarMVhost: possui parâmetros de entrada dos números de identificação da MV e do Servidor que esta será alocada. Ele realiza a alocação caso a quantidade de MIPS livre no Servidor seja maior que a requisitada pela MV. Após isso, ele diminui a quantidade de MIPS disponível no Servidor, preenche os campos correspondentes a alocação tanto na lista de MV como na lista de

Hosts. Retorna um valor booleano verdadeiro caso consiga fazer a alocação ou falso caso contrário.

- iniciarAlocacao: possui como parâmetro entrada o ID do método de alocação. Esta função deve chamar diretamente a função criada que realiza a alocação.
- respostaAlocacao: esta função cria uma lista com as alocações feitas. Esta lista possui uma divisão para MVs com alocação e sem alocação. Na lista de MVs com alocação fica registrado o ID do Servidor e a quantidade de MIPS disponível nele. A lista de sem alocação, registra somente o ID das MVs. Tem como retorno uma lista que possui as duas listas citadas.

5.2.3 Camada de Controle

A camada de Controle se encontra no diretório “controlador” e possui como único arquivo “hosts_mvs_controlador.php”. Este documento é responsável pelas seguintes funções:

- Receber a requisição Ajax vinda da camada de Visão.
- Instanciar os objetos das classes de Modelo e da classe de resposta da Visão.
- Chamar os métodos de criação de Hosts e MVs passando como parâmetros os dados vindos da requisição Ajax.
- Chamar o método de verificação de recursos que informará se é possível criar este Datacenter. Caso não seja possível deverá chamar o método que exibe erros pertencente a camada de Visão.
- Chamar o método que inicia a alocação na camada de Modelo e a seguir, o método que exibe os resultados vindos da camada de Modelo de forma a gerar a visualização final.

5.2.4 Camada de Visão – Resposta

Além dos outros arquivos da camada de Visão citados anteriormente, há um último arquivo responsável pela resposta ao cliente.

O arquivo “hosts_mvs_visao.php” foi implementado com os seguintes métodos:

- exibirErro: possui como parâmetro de entrada o ID do erro e através deste, retorna a mensagem de erro correspondente. A figura 12 mostra a mensagem de erro sendo exibida no cliente.

Figura 12 – Exemplo de mensagem de erro

Fonte: Criado pelo Autor(2015)

- exibirResultado: possui como parâmetro de entrada a lista de alocação criada na camada de Modelos e repassada pela camada de Controle. Através desta lista são criadas as tabelas de visualização de resposta. A tabela de MVs alocadas possui as informações de ID do Servidor, MVs alocadas, Quantidade de MVs alocadas, MIPS total do Servidor e MIPS livre do Servidor. A tabela de MVs sem alocação possui informação das MVs que estão desalocadas.

5.3 TESTES

Para realizar os Testes Unitários cada função foi testada separadamente das outras. No Frontend os estilos foram testados conforme criados. No Backend, as funções foram testadas através da passagem de parâmetros e verificação de retornos.

5.4 VALIDAÇÃO

Para validar a ferramenta, foi implementado o algoritmo First Fit. Para isto, foram feitas alterações nos seguintes documentos:

- “conteudo_visao.php”: foi incluído no rótulo de seleção de métodos a opção FCFS que referencia este método e atribuído valor 1.
- “alocacao_modelo.php”: foi adicionado a função iniciarAlocacao uma opção caso o valor 1 seja passado como parâmetro, para realizar esta função. Além disso, foi criada uma nova função chamada FF e implementado seu algoritmo. A figura 13 mostra as alterações feitas neste arquivo.

Figura 13 – Alterações do arquivo “alocacao_modelo.php”

```
164 /*****************************************************************************************
165 ********** FUNÇOES QUE DEVEM SER ALTERADAS PARA REALIZAR ALOCAÇÃO        ********************
166 *****************************************************************************************/
167
168     public function iniciarAlocacao($id_alocacao) {
169             switch ($id_alocacao) {
170                 case 1:
171                     return $this->firstFit();
172                     break;
173                 default:
174                     break;
175             }
176
177     }
178
179     private function firstFit(){
180         //mvs_alocadas(idHost(idMv(mipsMv)))
181         //mvs_desalocadas(idMv)
182
183
184         for ($idMv=0; $idMv < count($this->mvs); $idMv++) {
185             for ($idHost=0; $idHost < count($this->hosts); $idHost++) {
186                 //var_dump($this->alocarMvHost($idMv, $idHost));
187                 if($this->alocarMvHost($idMv, $idHost)){
188                     break;
189                 }
190             }
191         } unset($idMv, $idHost);
192
193         return $this->respostaAlocacao();
194     }
```

Fonte: Criado pelo Autor (2015)

Após a criação do algoritmo, foram gerados os cenários necessários para validar a ferramenta. Para isto, foi necessário iniciar o WAMPServer, abrir o navegador Google Chrome e digitar na barra de endereços a palavra “localhost” e então clicado na pasta “Alocador”.

5.4.1 Cenário 1 – Servidor sem MVs alocadas

Para demonstrar este cenário em que há um Servidor sem Máquinas Virtuais alocadas foram preenchidos os campos iniciais do formulário de Datacenter, da seguinte forma:

- Campo de quantidade de Hosts: 2
- Campo de quantidade de MVs: 1
- Tipo de Alocação: FCFS

A Figura 14 representa os dados que foram preenchidos nos inputs (valores de entrada) deste formulário.

Figura 14 – Valores de entrada do formulário inicial do cenário 1

Fonte: Criado pelo Autor(2015)

Depois de preenchido e pressionado o botão para prosseguir, foi gerado dois novos formulários baseados nos valores. Nos novos formulários exibidos foram preenchidos os campos de Hosts e MVs todos com os valores 100. A Figura 15 demonstra o preenchimento dos campos.

Figura 15 – Campos preenchidos dos formulários de Hosts e MVs

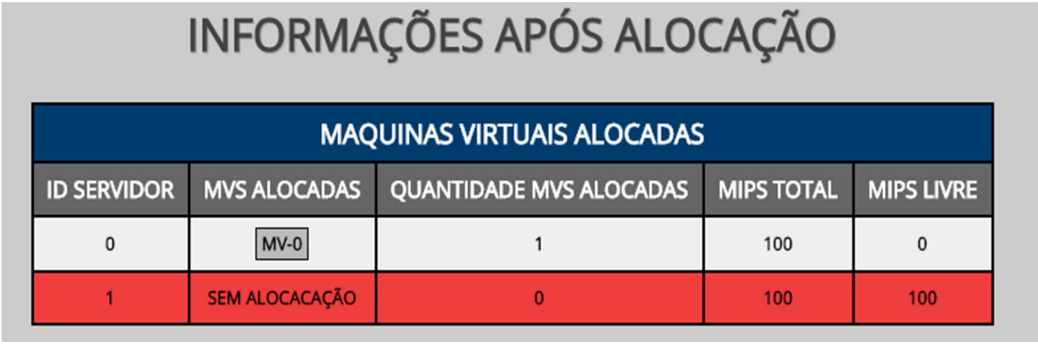
Fonte: Criado pelo Autor(2015)

Desta forma, o segundo Servidor não possuirá Máquinas Virtuais para alocar. O resultado esperado após pressionar o botão para realizar a alocação é:

- Host 0: Mv-0 alocada, 1 MV alocada, Total de MIPS 100 e 0 MIPS livre.
- Host 1: Sem alocação, 0 MVs alocadas, Total de MIPS 100 e 100 MIPS livres.

A Figura 16 demonstra este resultado fornecido através da ferramenta. Este resultado foi igual ao esperado conforme o método de alocação prevê.

Figura 16 – Resultado do Cenário 1



Fonte: Criado pelo Autor(2015)

5.4.2 Cenário 2 - Uma MV não alocada

Para este cenário, procurou-se fazer com que uma das Máquinas Virtuais ficasse desalocada de todos os Servidores. Foi seguida a mesma rotina de preenchimento dos formulários conforme demonstrado no tópico anterior, porém com as seguintes modificações:

- Quantidade de Hosts: 3
- Quantidade de MVs: 4
- Tipo de Alocação: FCFS

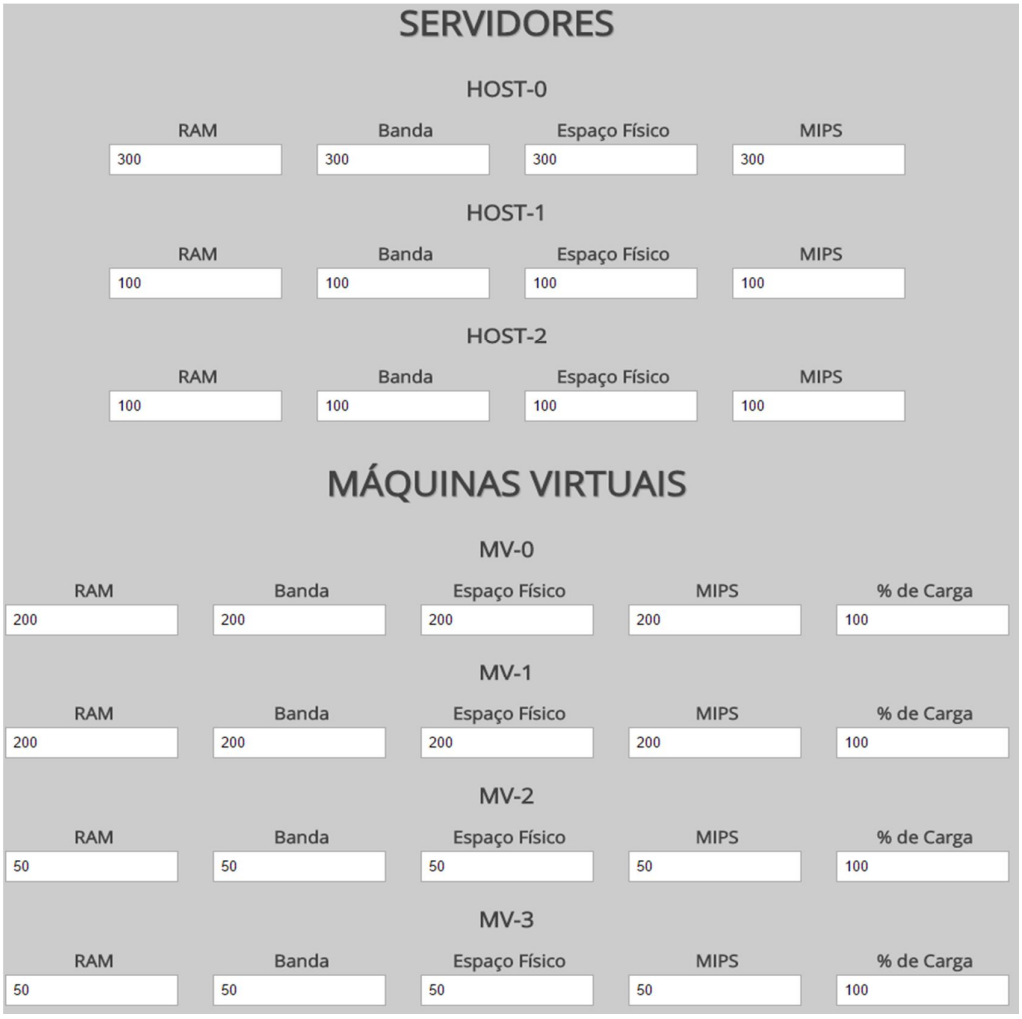
A Figura 17 demonstra as modificações feitas no formulário do Datacenter.

Figura 17 – Preenchimento dos campos do Datacenter para o cenário 2

Fonte: Criado pelo Autor (2015)

Foram modificados também os formulários seguintes. A figura 18 demonstra estas modificações no preenchimento.

Figura 18 – Preenchimento dos campos de Hosts e MVs para o cenário 2

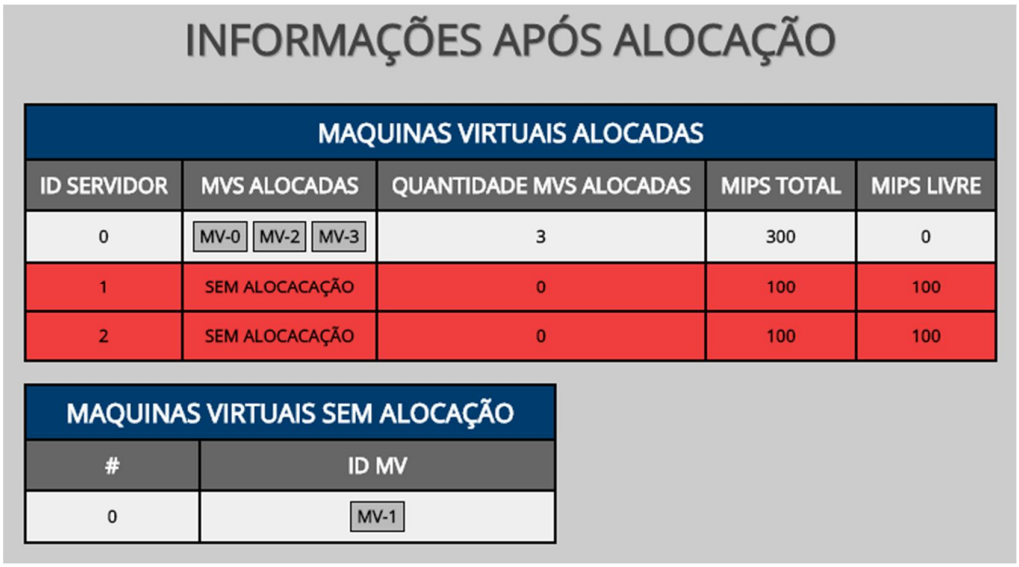


Fonte: Criado pelo Autor (2015)

Da mesma forma, foi gerado um resultado, o resultado esperado pela realização do algoritmo seria que a “MV-1” estivesse desalocada neste Datacenter e que o Servidor 0 contivesse a MV-0, MV-2 e MV-3 alocadas. Além disso, os Servidores 1 e 2 estivessem sem nenhuma MV alocada.

A Figura 19 demonstra o resultado gerado pela ferramenta. Esta se comportou de modo esperado e exibiu os resultados conforme o algoritmo prevê.

Figura 19 – Resultado do Cenário 2 gerado pela ferramenta



Fonte: Criado pelo Autor (2015)

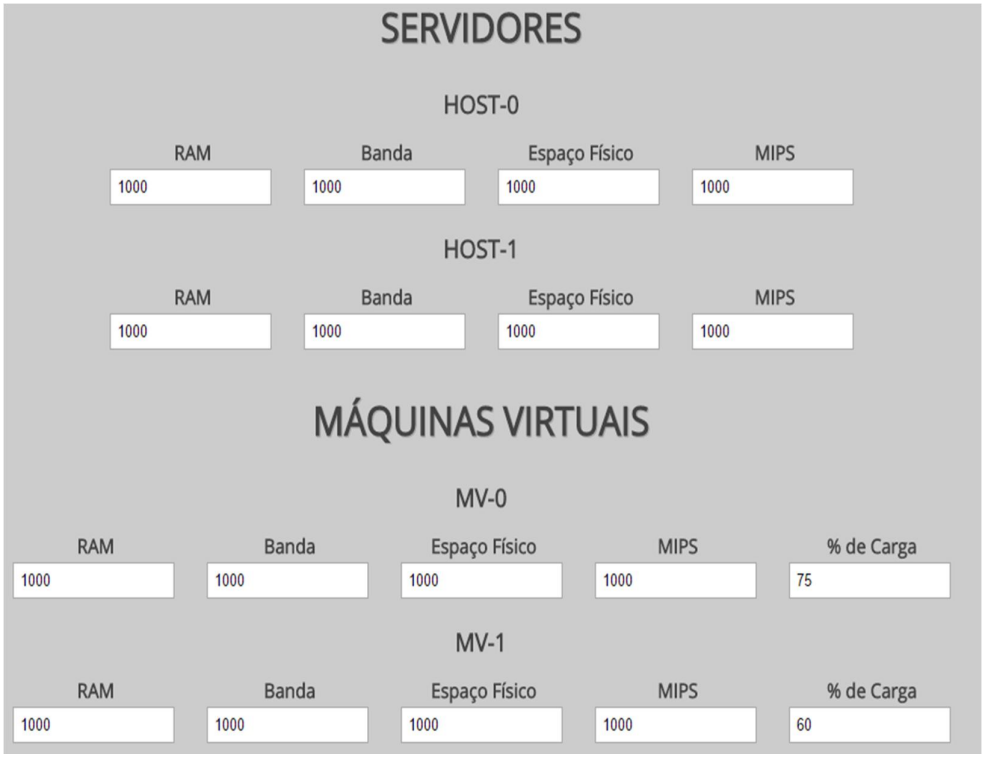
5.4.3 Cenário 3 – Todos os Servidores possuem MVs alocadas

Para este cenário em que todos os Servidores possuam MVs alocadas, também foi seguida a mesma rotina de preenchimento dos anteriores, porém com as seguintes modificações:

- Quantidade de Hosts: 2
- Quantidade de MVs: 2
- Tipo de Alocação: FCFS

Foram modificados também os formulários seguintes. Nos campos de Hosts e MVs todos os valores foram preenchidos com 1000, exceto a porcentagem de carga que foi preenchida com 75 para a MV-0 e 60 para MV-1. A Figura 20 demonstra estas modificações no preenchimento.

Figura 20 - Preenchimento dos campos de Hosts e MVs para o cenário 3

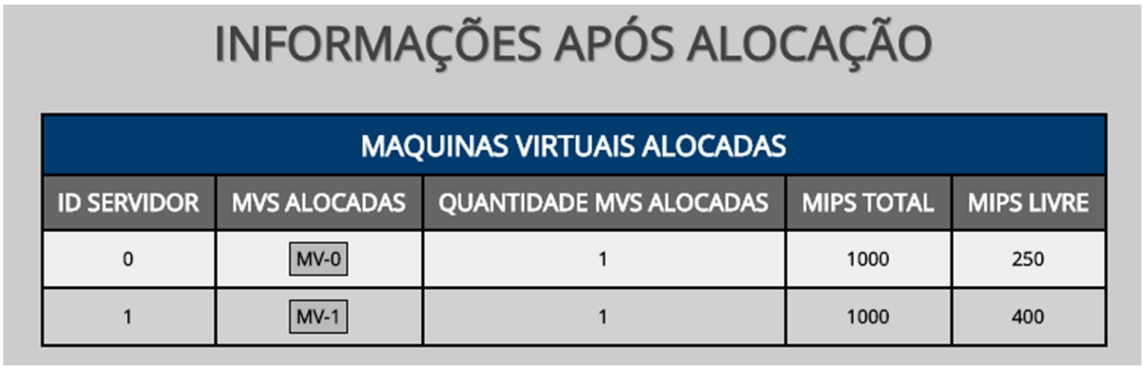


Fonte: Criado pelo Autor (2015)

Também foi gerado um resultado da ferramenta, e o resultado esperado pela realização do algoritmo seria que a MV-0 estivesse alocada no Host-0 e a MV-1 estivesse alocada no Host-1.

A Figura 21 demonstra o resultado gerado pela ferramenta. Esta se comportou de modo esperado e exibiu os resultados conforme o algoritmo prevê.

Figura 21 – Resultado do cenário 3 gerado pela ferramenta



Fonte: Criado pelo Autor (2015)

6 CONSIDERAÇÕES FINAIS

A computação em Nuvem possui grande potencial para o futuro e já está sendo adotada por diversas empresas. Faz-se importante então, o estudo de técnicas que envolvam melhorias de seus processos. Uma dessas técnicas, a alocação de recursos, gera vantagens de economia para os datacenters de uma Nuvem.

Mediante a criação da ferramenta Web os desenvolvedores poderão testar de forma ágil e simples novos conceitos para alocação de Máquinas Virtuais nos Servidores da Nuvem utilizando a dimensão de MIPS.

Considerando o potencial desta ferramenta, pode-se ampliá-la para realizar o trabalho com todas as dimensões possíveis de Máquinas Virtuais e Servidores a fim de estimar de forma mais completa a alocação dos recursos. E, inclusive, poderá ser adicionada uma forma de testar a alocação de tarefas nas Máquinas Virtuais seguindo os mesmos conceitos deste framework.

Verificou-se que a nuvem representa um novo modelo de utilização da tecnologia que é vista como um novo modelo de negócios criado para atender à necessidade de redução de custos. Planejando de forma adequada é possível incorporar as técnicas criadas de forma a torná-la mais eficiente e ainda menos custosa.

REVISÃO BIBLIOGRÁFICA

AGOSTINHO, P.; **Virtualização em SAP**, Universidade Lusófona de Humanidades e Tecnologias, 6º SOPCOM, Lisboa, Portugal, 2009.

ARMBRUST, M et al. **Above the Clouds:** A Berkeley View of Cloud Computing. Berkeley, EECS Department, University of California, 2009.

BARTIÉ, A. **Garantia da Qualidade de Software**. Rio de Janeiro: Elsevier, 2002.

BASHAM, B.; SIERRA K.; BATES B. **Head first servlets & jsp**. 2ed, O'Reilly, 2008.

BUYENS, Jim. **Aprendendo MySQL e PHP**. 1ed. São Paulo, 2002.

BUYYA, R.; BELOGLAZOV, A; ABAWAJY, J. **Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing**. Future Generation Computer Systems, v. 28, n. 5, p. 755 – 768, 2012.

CALHEIROS, R. et al. **CloudSim**: a Toolkit for Modeling and Simulation of Cloud Computing Environments and Evaluation of Resource Provisioning Algorithms. Software: Practice and Experience (SPE), Nova York, v. 41, n. 1, p. 23-50, 2011.

CALHEIROS, R.; RANJAN, R.; BUYYA, R.; **Modeling and Simulation of Scalable Cloud Computing Environments and the CloudSim Toolkit**: Challenges and Opportunities, The 2009 International Conference on High Performance Computing and Simulation, HPCS, pp:1-11, 2009.

CARISSIMI, A.; **Virtualização**: Princípios Básicos e Aplicações, ERAD 2009, SBC, Caxias do Sul, 2009.

CASAVANT, T.; KUHL, J. **A Taxonomy of Scheduling in General-purpose Distributed Computing Systems**. IEEE Transactions on Software Engineering, vol. 14, pp. 141-154, 1988.

ERAS, M. **Cloud Computing:** Nova arquitetura da TI, Rio de Janeiro, Editora Brasport, 2012

FAYAD, M.; SCHMIDT, D. **Object-oriented application framework**. Communications of the ACM, v. 40, p. 32-38, 1997.

FERREIRA, E; EIS, D. HTML5 – Curso W3C Escritório Brasil. Disponível em: http://www.w3c.br/pub/Cursos/CursoHTML5/html5-web.pdf Acesso em: 22 de setembro de 2015.

FOX, C. **Introduction to software engineering design**: processes, principles, and patterns with uml. 1ed. Boston: Pearson, 2006.

FRANCO, P. B. **Escalonamento de Tarefas em Ambiente de Simulação de Grid Computacional**. Dissertação de Mestrado em Ciência da Computação, Universidade Estadual paulista Júlio de Mesquita Filho, UNESP, 2011, 100p.

GONÇALVES, E. **Desenvolvendo Aplicações Web com JSP Servlets, JavaServer Faces, Hibernate, EJB3 Persistence e Ajax**. p.141, Editora Ciência Moderna Ltda., 2007.

HYSER, C.; et al. **Autonomic virtual machine placement in the data center**. Technical Report HPL-2007-189, HP Labs, Palo Alto CA, 2007.

jQuery; **Write less, do more**. Disponível em: <www.jquery.com> Acesso em: 20 de setembro de 2015.

KABIR, J. **Apache Server 2, a Bíblia**. 1ed, Campus, Rio de Janeiro, 2002.

LIU, S.; LIANG, Y.; Brooks, M. **Eucalyptus**: a web serviceenabled e-infrastructure. In CASCON '07: Proceedings of the 2007 conference of the center for advanced studies on Collaborative research, pages 1–11, New York, NY, USA, 2007.

MASSOL, V; HUSTED, T. **Junit em Ação**. Rio de Janeiro: Ciência Moderna, 2005.

MENASCÉ, D.: **Virtualization**: Concepts, Applications, and Performance Modeling. George Mason University, EUA, 2005.

MISHRA, M.; SAHOO, A. **On theory of VM placement**: Anomalies in existing methodologies and their mitigation using a novel vector based approach. In IEEE 4th International Conference on Cloud Computing, pages 275–282. IEEE, 2011.

MORRISON, M. **Use a Cabeça Javascript**. 640p. São Paulo: Alta Books, 2008.

MORRISON, M. **Use a Cabeça! JavaScript**. p.4, AltaBooks, Rio de Janeiro, 2008.

NEMETZ, R. **Otimizando o Fluxo de Tarefas em Sistemas Distribuídos de Impressão:** um Algoritmo de Escalonamento Dinâmico Não Preemptivo Baseado em Mecanismo de Previsão. Dissertação de Mestrado, Programa de Pós-Graduação em Ciência da Computação, p. 79-81, PUCRS, 2011.

NIST. **The NIST Definition of Cloud Computing**, NIST, 2011.

NURMI D. et al. **The eucalyptus open-source cloud-computing system**, Proceedings of the 2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid, Volume 00, 2009.

POPEK, G.; GOLDBERG, R. **Formal requirements for virtualizable third generation architectures**. Communications of the ACM, 17(7):412– 421, 1974.

SABAHI, F. **Cloud computing security threats and responses. Communication Software and Networks (ICCSN)**, 2011 IEEE 3rd International Conference on May 2011

SEIDEN, S; **On the online bin packing problem**. J. ACM, 49(5):640–671, 2002.

SHANKAR, A. **Virtual Machine Placement in Computing Clouds**. Technical Report. 23 p, 2010.

SINGH, A.; KORUPOLU, M.; MOHAPATRA, D. **Server-storage virtualization**: Integration and load balancing in data centers. In ACM/IEEE Conference on Supercomputing, 12p. IEEE Press, 2008

SOARES, W. **AJAX (Asynchronous Javascript And XML)**: guia prático para Windows. São Paulo, Ética, 2006.

SOUSA, F.; MOREIRA, L.; MACHADO, J. **Computação em Nuvem**: Conceitos, Tecnologias, Aplicações e Desafios. Fortaleza. 2009.

TANEMBAUM, A.; WOODHULL, A. **Sistemas Operacionais, Projeto e Implementação**. Tradução João Tortello. 3 ed. Porto Alegre: Bookman, 2008, 992p.

TAURION, C. **Computação em Nuvem:** Transformando o mundo da tecnologia da informação, Rio de Janeiro, Brasport, 2009

W3C. **Cascanding Style Sheets**. Disponível em: <http://www.w3.org/Style/CSS>. Acesso em: 15 de setembro de 2015.

ZANDSTRA. M. **Entendendo e Dominando o PHP**. 1ed. São Paulo: Digerati Books, 2006.

ZHANG, Q.; CHENG, L.; BOUTABA, R. Cloud computing: state-of-the-art and research challenges. **Journal of Internet Services and Applications**, v. 1, n. 1, p. 7–18, 2010.