UNIOESTE – Universidade Estadual do Oeste do Paraná

CENTRO DE CIÊNCIAS EXATAS E TECNOLÓGICAS

Colegiado de Ciência da Computação

Curso de Bacharelado em Ciência da Computação

Implementação de mecanismos automáticos de elasticidade

Mauriverti da Silva Junior

CASCAVEL

2015

MAURIVERTI DA SILVA JUNIOR

IMPLEMENTAÇÃO DE MECANISMOS AUTOMÁTICOS DE ELASTICIDADE

Monografia apresentada como requisito parcial para obtenção do grau de Bacharel em Ciência da Computação, do Centro de Ciências Exatas e Tecnológicas da Universidade Estadual do Oeste do Paraná - Campus de Cascavel

Orientador: Guilherme Galante

CASCAVEL

2015

MAURIVERTI DA SILVA JUNIOR

IMPLEMENTAÇÃO DE MECANISMOS AUTOMÁTICOS DE ELASTICIDADE

Monografia apresentada como requisito parcial para obtenção do Título de *Bacharel em Ciência da Computação*, pela Universidade Estadual do Oeste do Paraná, Campus de Cascavel, aprovada pela Comissão formada pelos professores:

Prof. Guilherme Galante
Colegiado de Ciência da Computação,
UNIOESTE

Prof. Marcio Seiji Oyamada
Colegiado de Ciência da Computação,
UNIOESTE

Prof. Aníbal Mantovani Diniz
Colegiado de Ciência da Computação,
UNIOESTE

Cascavel, 10 de março de 2016

Agradecimentos

Agradeço a todos que fizeram parte da minha vida durante a graduação, principalmente a toda minha família e minha namorada Jessica Lenes dos Reis da Silva, por todo o carinho, apoio e incentivo.

Aos meus amigos Cezar Leandro Manica, Diego Henrique Pagani, Gean Carlo Peixoto, Julio Cesar Lazzarim e Thiago Junior Vacari por todas as festas, trabalhos, alegrias e sofrimentos compartilhados durante os anos de graduação.

Ao meu orientador Guilherme Galante pelo auxílio, apoio e paciência durante a realização deste trabalho.

A toda banca avaliadora pelas dicas e correções que aprimoraram o conteúdo deste trabalho.

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas e Siglas

API	*Application Programming Interface*
CaaS	*Communication as a Service*
CPU	*Central Processing Unit*
DBaaS	*Data Base as a Service*
DevaaS	*Development as a Service*
Dom0	*Domain 0*
DomU	*Unprivileged Domain*
GB	Gigabyte
IaaS	*Infrastructure as a Service*
ID	Identificador
MB	Megabyte
MV	Máquina Virtual
NIST	*National Institute of Standards and Technology*
PaaS	*Plataform as a Service*
RAM	*Random Access Memory*
SaaS	*Software as a Service*
SGBD	Sistema Gerenciador de Banco de Dados
SO	Sistema Operacional
TI	Tecnologia da Informação
VCPU	*Virtual Central Processing Unit*

Sumário

Resumo

A elasticidade é a principal característica da computação em nuvem, diferenciando-a de outros sistemas distribuídos como computação em grade e clusterização. Ela é responsável pela dinamicidade de uma máquina virtual, fazendo com que seus recursos, não sejam fixos e possam ser aumentados ou reduzidos, dependendo da carga de trabalho, este tipo de elasticidade é conhecido como vertical, já no tipo horizontal, este dinamismo fica por parte do número de máquinas virtuais disponíveis para a execução de determinado processo que trabalhe de maneira distribuída. A Elasticidade horizontal é a mais encontrada em aplicativos de gerenciamento de nuvem, sendo que a implementação do tipo vertical está em uma minoria destes sistemas, principalmente se tratando de nuvem privada. Neste contexto, o propósito deste trabalho foi desenvolver uma ferramenta que permita utilizar a elasticidade vertical e horizontal de maneira automática, realizando este processo de alocação/liberação de recursos, partindo de regras previamente definidas e analisando a quantidade de recursos que cada máquina virtual está usando. Por fim são apresentados resultados de alguns testes de estresse realizados, em CPU e memória, durante períodos de tempo curtos e longos, além de alguns cuidados que devem ser levados em conta na hora de escolher o tipo de elasticidade e a quantidade de recursos alocado/liberado dependendo do tipo de aplicação que estará em execução nas máquinas virtuais.

Palavras-chave: Computação em nuvem, Elasticidade, Monitoramento de Máquinas Virtuais.

Capítulo 1
Introdução

Segundo *National Institute of Standards and Technology* (NIST) [1], a computação em nuvem, ou *Cloud computing*, pode ser definido como um modelo que permite acesso a recursos computacionais (redes, servidores, armazenamento, aplicações e serviços) sob demanda, através da rede, que podem ser rapidamente provisionados e liberados, ou seja, utilizar recursos de outros computadores de maneira simples e eficiente através de conexão de rede, sem se preocupar com a sua localização ou configuração.

Normalmente, ela é disponibilizada através do modelo de pagamento pelo uso (*pay-per-use*), assim, qualquer pessoa pode contratar o serviço de alguma empresa, como Amazon AWS ou Windows Azure e pagar apenas pelo que for realmente utilizado (memória, processadores, transmissão de rede), este é o modelo conhecido como nuvem pública [2]. Porém, não são todas as empresas que confiariam dados importantes de seus negócios a outras empresas, e mesmo assim gostariam de ter a disponibilidade e facilidade da *Cloud Computing*. Nestes casos é possível utilizar uma nuvem privada, em que o centro de dados fica interno a organização, assim, apenas ela tem acesso as informações disponíveis.

Toda esta estrutura parte do princípio da virtualização, onde vários ambientes virtuais podem ser executados sobre o mesmo hardware físico. Estes ambientes virtuais podem ser disponibilizados aos usuários, a qualquer momento e em qualquer quantidade [1] e, caso necessitem, é possível adicionar ou remover recursos quando há poder computacional em falta ou excesso, sem causar qualquer interrupção [3]. Esta funcionalidade é chamada de elasticidade.

A Elasticidade é a capacidade de se adquirir ou liberar recursos de acordo com a demanda, e pode ser dividida em duas classes: horizontal e vertical [4], [5].

A elasticidade horizontal, também conhecida como replicação, acontece quando há uma grande demanda e a máquina virtual (MV) já está sobrecarregada, então uma nova máquina virtual é adicionada, ou replicada, para dividir a esta carga. Já a elasticidade vertical, consiste em alocar mais recursos para a MV, como memória e processadores, para assim atender todas as requisições.

Atualmente, diversas soluções automatizadas de elasticidade são fornecidas pelos principais provedores públicos de nuvem, tais como, Amazon, Azure, Rackspace. Estas soluções, em geral, empregam elasticidade horizontal e utilizam balanceadores de carga para distribuir a demanda entre as diversas máquinas virtuais [4], [5], [6].

Porém, ao analisar os sistemas de gerenciamento de nuvens privadas, entre eles, o OpenNebula, o Eucalyptus e o OpenStack, é possível notar que a maioria destes, não implementam mecanismos de elasticidade automáticos como os propostos pelos provedores públicos [5], [6]. A solução oferecida por estes é, geralmente, a possibilidade de instanciação manual de MVs.

Considerando o exposto, este trabalho tem como objetivo proporcionar uma solução que adeque os recursos disponíveis para uma máquina virtual à sua demanda em tempo real, para assim aproveitar melhor os recursos da máquina física em que ela está alocada, ou seja, apresentar um sistema que controle a elasticidade tanto vertical quanto horizontal, automaticamente, de maneira reativa.

Como resultados, espera-se que o software desenvolvido consiga identificar pontos de sobrecarga ou de ociosidade da MV, através de monitores, e tome uma ação a partir de regras definidas previamente (mecanismo regra-condição-ação). Estas regras utilizam limiares pré-definidos baseados na carga atual sobre a capacidade total de memória e núcleos, obtidos a partir dos monitores, e executam rotinas de alocação de recursos quando o limite superior for ultrapassado ou liberação de recursos, caso o limite inferior seja ultrapassado, sempre respeitando limites máximos e mínimos também pré-definidos.

Este sistema será validado através de softwares que estressem o sistema tanto em CPU quanto em memória, com duração curta e longa, para assim analisar desde quando a necessidade de gerar uma alocação/remoção é detectada até o início do processo e o tempo de alocação/liberação de recursos, tanto na abordagem vertical quanto na horizontal.

Capítulo 2
Computação em nuvem

A Computação em Nuvem está se tornando uma das palavras chaves da indústria de Tecnologia da Informação. A nuvem é uma metáfora para a Internet ou infraestrutura de comunicação entre os componentes arquiteturais, baseada em uma abstração que oculta à complexidade da infraestrutura [7]. Vaquero [8] avaliou mais de 20 definições de computação em nuvem, para propor uma unificada. Segundo ele "Nuvens são grandes conjuntos de recursos virtualizados (como *hardware*, plataformas de desenvolvimento e/ou serviços), acessíveis e de fácil uso. Estes recursos podem ser dinamicamente reconfigurados para se ajustar a variação de carga, permitindo também uma otimização em seu uso. Este conjunto de recursos é tipicamente explorado pelo modelo pague-pelo-uso que é garantido pelos Provedores de Infraestrutura através de acordos de serviço customizados".

Mell e Grance [9], apresentam o modelo computacional de nuvem baseado em um conjunto de servidores e toda a infraestrutura necessária para seu funcionamento (como eletricidade, refrigeração, conexões de rede e armazenamento), que são compartilhados entre os usuários. Este compartilhamento é feito através do uso de virtualização, que permite que várias máquinas virtuais possam ser executadas sobre o mesmo hardware físico, permitindo a otimização do uso destes recursos.

O uso da virtualização permite que os recursos da nuvem sejam fornecidos de modo escalável, dinâmico e de acordo com as demandas do usuário [6]. Portanto os recursos podem ser alocados ou liberados rapidamente, fazendo com que o usuário aparente possuir recursos ilimitados, podendo se apropriar deles a qualquer momento e em qualquer quantidade [1].

Pode-se classificar uma nuvem de acordo com o seu modelo de implantação e seu modelo de serviço. Os modelos de implantação são abordados na Seção 2.1 e os modelos de serviço na Seção 2.2

2.1. Modelos de implantação

Há atualmente, quatro modelos de implantação da computação em nuvem. Sua nomenclatura depende da necessidade, disponibilidade e segurança exigidos pelos contratantes. São eles: Pública, Privada, Comunitária e Híbrida [2].

As nuvens públicas são aquelas executadas por terceiros, podem ser gratuitas ou pagas no formato de pagamento pelo uso (*pay-per-use*) [10], as aplicações de diversos usuários compartilham uma mesma infraestrutura, o que pode parecer ineficiente a princípio. Porém, se a implementação de uma nuvem pública considera questões fundamentais, como desempenho e segurança. A existência de outras aplicações sendo executadas na mesma nuvem permanecem transparentes tanto para os prestadores de serviços como para os usuários [11]. Nuvens públicas tem como característica a alta disponibilidade, porém podem pecar no quesito de segurança, que fica completamente a cargo do provedor do serviço.

Por outro lado, as nuvens privadas são construídas para o uso exclusivo de uma organização, para atender suas necessidades internas. Diferentemente da nuvem pública, todo o *data center* é gerenciado, localizado e operado de dentro da organização, porém é possível contratar uma empresa terceirizada para realizar tal tarefa. A disponibilidade é reduzida em função da sua alta segurança, o que é essencial para as empresas manterem segredos de negócio.

A Nuvem comunitária é estruturada por uma nuvem compartilhada por várias organizações que dividem interesses, como por exemplo segurança, política, missão, etc. Ela pode ser gerenciada por uma ou várias das organizações, por uma empresa terceirizada, ou ainda por uma junção destas e o *data center* pode ser interno ou externo a estas organizações. Neste modelo, sacrifica-se um pouco da segurança pela comodidade de se compartilhar informações úteis entre os parceiros ou até mesmo reduzir custos das empresas envolvidas.

Por fim, o modelo de nuvem híbrida é uma mescla de dois ou mais modelos anteriores, que aproveita os benefícios de cada uma. É utilizada, normalmente, em casos que a nuvem privada ou comunitária não suporta a demanda necessária, podendo então migrar máquinas virtuais da nuvem privada para a pública, mantendo o serviço em funcionamento durante as dificuldades encontradas.

2.2. Modelos de serviço

Além dos modelos de implantação, há também os modelos de serviços, que são divididos de acordo com a capacidade de abstração dos recursos oferecidos pela nuvem. Há vários modelos: software como serviço (SaaS); plataforma como serviço (PaaS); infraestrutura como serviço (IaaS); desenvolvimento como serviço (DevaaS); comunicação como serviço (CaaS); banco de dados como serviço (DBaaS); entre outros, porém todos eles podem ser englobados apenas nos três primeiros [1], [12].

Estes modelos principais estão separados em camadas, onde os serviços de uma camada superior podem ser compostos por serviços da camada inferior subjacente [2], desta maneira um software necessita de uma plataforma, que está executando sobre uma infraestrutura e o administrador da rede libera o acesso destes serviços aos clientes, dependendo do tipo de serviço contratado.

2.2.1. Software como serviço (SaaS)

Neste modelo de serviço, o cliente tem acesso apenas a um software especifico e nada além disso, normalmente utilizando apenas um navegador web ou uma *Application Programming Interface* (API). Hoje em dia, há muitos exemplos de sistemas que utilizam esta tecnologia, como gerenciadores de e-mail, editores de texto e planilha eletrônica [13], [14], ou qualquer outro software, sem que o usuário se preocupe com instalação, manutenção, compatibilidade e atualização destes sistemas, tudo isto fica por parte do provedor.

O SaaS também pode ajudar empresas que em vez de comprarem uma licença de software para cada desktops e servidores para aplicações, ela pode obter a mesma função utilizando os serviços em nuvem de um provedor através da internet ou alguma rede de comunicações [12], diminuindo os custos tanto nos softwares, como na necessidade de comprar um hardware potente.

2.2.2. Plataforma como serviço (PaaS)

No nível de plataforma, a visão do cliente é ampliada trazendo, além do uso de softwares, a capacidade de desenvolvê-los. Também é possível testá-los e armazená-los na rede, sem se preocupar com o gerenciamento de sistemas operacionais, correções e atualizações, cargas de processamento, disponibilidade e balanceamento de carga, entre outros [2]. O usuário não tem

acesso a infraestrutura, como sistema operacional (SO), rede e armazenamento, mas tem controle para configurar as aplicações implantadas.

As aplicações desenvolvidas podem conter certas limitações, dependendo do provedor, como o tipo de software que pode ser desenvolvido e limitações no tipo de sistemas de gerenciamento de banco de dados (SGBDs) que pode ser utilizado [15]. Alguns exemplos desta plataforma são o Google App Engine [16] e o Microsoft Azure [17].

2.2.3. Infraestrutura como serviço (IaaS)

O modelo IaaS é o mais flexível pois permite ao usuário configurar exatamente o que precisa, incluindo sistema operacional, aplicações, armazenamento e em alguns casos selecionar componentes de rede, como *firewalls* [1]. Segundo Souza et al. [15] o termo IaaS se refere a uma infraestrutura computacional baseada em técnicas de virtualização de recursos de computação, pois torna mais fácil e acessível o fornecimento de recursos, tais como servidores, rede e armazenamento. Exemplos conhecidos são o Amazon EC2, o Eucalyptus, o OpenNebula, entre outros.

Algumas características deste modelo são uma interface única para administração da infraestrutura, API para interação com *hosts*, *switches*, balanceadores de carga, roteadores e o suporte para a adição de novos equipamentos de forma simples e transparente [2]. Esta adição de equipamentos, pode também ser entendida como novas máquinas virtuais e até mesmo recursos, caso haja necessidade, tudo isto em tempo de execução, esta funcionalidade chama-se **elasticidade** e é uma das principais características da Computação em Nuvem. Mais detalhes sobre elasticidade são descritos no Capítulo 3.

Capítulo 3
Elasticidade

Elasticidade é uma característica fundamental da computação em nuvem [18]. Ela pode ser definida como a capacidade de um sistema se adaptar a mudanças na demanda, adicionando ou removendo recursos de maneira dinâmica [19]. Estes recursos podem incluir desde CPU, memória e armazenamento, até máquinas virtuais completas.

A Figura 3.1, adaptada de Righi [5], mostra a diferença entre um ambiente estático (a) e um elástico (b) em uma máquina virtual sendo executada sozinha em um servidor físico, sendo que 100% representa a quantidade máxima de CPU disponível no servidor. No ambiente estático a alocação de recursos é fixa e determinada antes da inicialização da máquina virtual. Ela tenta sempre reservar recursos para os picos mais altos que a demanda pode alcançar, para que assim nenhuma aplicação seja afetada ou solicitação não seja atendida. Entretanto, esta carga estaria sempre alocada para esta MV, mesmo que ela não chegue a usar todos os recursos. Já no ambiente elástico, os recursos apenas são alocados quando há a necessidade, caso contrário, eles são liberados para que outras máquinas virtuais possam utilizá-lo

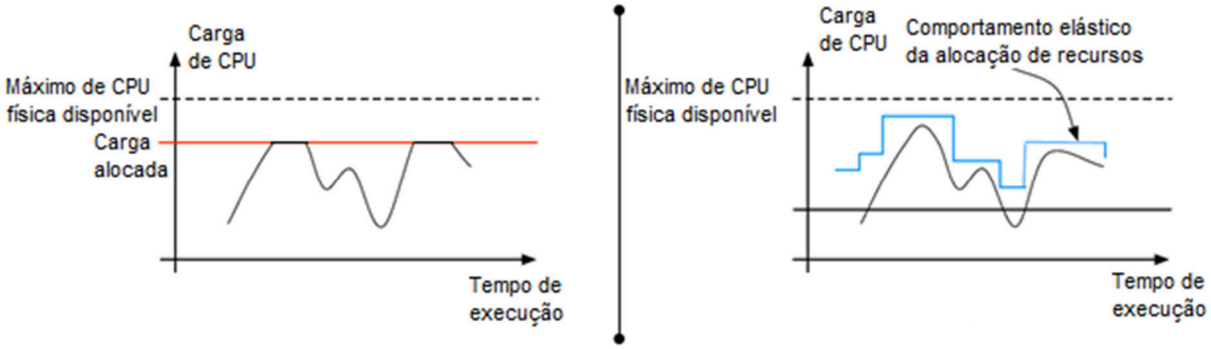

Figura 3.1: Exemplo de carga por tempo em uma máquina sem elasticidade (a) e com elasticidade (b).

Estimar com precisão a quantidade de recursos que será alocada não é uma tarefa fácil. Se poucos recursos forem alocados, poderá haver momentos em que a carga ultrapasse sua capacidade, comprometendo a execução do sistema. Por outro lado, se houver alocação excessiva de recursos, em alguns momentos poderá haver muito poder computacional ocioso.

Armbrust [20] afirma que, um *Data Center* utiliza, normalmente, entre 5% e 20% de sua capacidade, entretanto em momentos de pico, a carga de trabalho pode aumentar de 2 a 10 vezes.

A elasticidade surge para resolver este tipo de problema, pois é possível alocar dinamicamente os recursos de acordo com a demanda. Caso uma máquina virtual necessite de poder computacional adicional, mais recursos podem ser alocados em tempo de execução e por outro lado, caso haja recursos ociosos pode-se liberá-los para que sejam utilizados por outras máquinas.

Galante e Bona [21] apresentam duas políticas de como a alocação pode ser feita: manual, onde o usuário insere e remove os recursos manualmente; e automática, onde isto é feito automaticamente durante a execução. Esta segunda ainda pode ser dividida em dois mecanismos de automação: reativo, em que uma ação é executada a partir de regras pré-configuradas (mecanismo regra-condição-ação) pelo usuário ou programador; e preditivo, em que são usadas heurísticas e técnicas matemáticas/analíticas, para antecipar a demanda e assim poder fazer as alocações e liberações necessárias.

A capacidade de expandir e contrair, elasticamente, a base de recursos é interessante tanto para o provedor quanto para o usuário final da nuvem [6]. Neste primeiro por melhorar o uso dos recursos de computação, fornecendo economia de escala e permitindo que mais usuários possam ser atendidos simultaneamente, uma vez que quando um usuário libera algum recurso, ele pode ser instantaneamente alocado por outro. No segundo ela pode ser utilizada por diversos fins, como complemento de recursos locais [22], [23], redução de custos [24], [25], economia de energia elétrica [25], entre outros.

A maneira como é implementado este aumento e diminuição de recursos a divide em três tipos: Migração, Horizontal, Vertical [5], [21], apresentados respectivamente nas Seções 3.1, 3.2 e 3.3. Na Seção 3.4 será apresentado as soluções que já existem, tanto em hipervisores, quanto em gerenciadores de máquinas virtuais.

3.1. Migração

Righi [5] afirma que a migração é o tipo mais simples de elasticidade, ele consiste, basicamente, em migrar uma máquina virtual que está funcionando em um servidor para outro servidor que melhor se adeque ao tipo de aplicação que a MV contém. Máquinas virtuais

possuem a característica de isolamento e *hypervisors* como o XEN e o KVM. A transferência entre máquinas físicas ocorre com um nível de desempenho aceitável, fazendo com que o tempo de inatividade seja relativamente baixo.

A Figura 3.2 apresenta um exemplo simples de migração. Em (a) temos dois servidores físicos, o primeiro (Servidor 1) com duas máquinas virtuais, e o segundo, (Servidor 2) com apenas uma. Neste caso, é possível migrar a máquina virtual contida no Servidor 2, para o Servidor 1, assumindo que ele possua recursos disponíveis suficientes. Esta migração poderia ser para aumentar o poder de processamento da MV do Servidor 2, já que o Servidor 1 possui um *clock* superior, ou também para economizar energia, já que após a migração ser concluída, como exposto em (b), o Servidor 2 não tem mais utilidade, podendo então ser desligado.

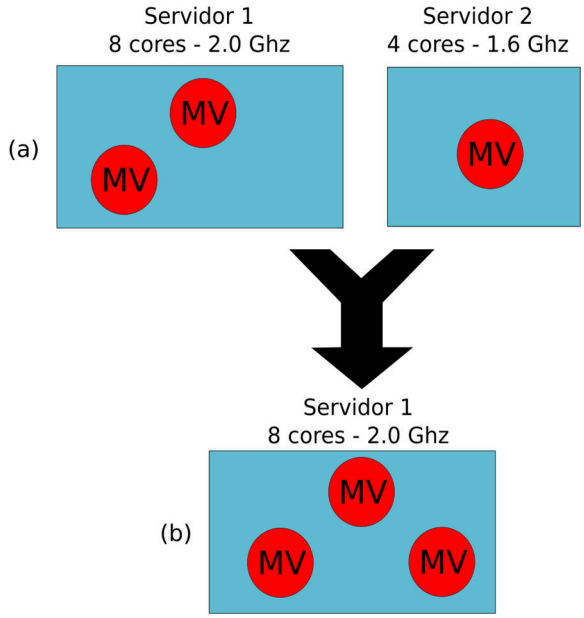

Figura 3.2: Exemplo de migração em nuvem de dois servidores físicos para um.

3.2. Horizontal

A elasticidade do tipo horizontal, é também conhecida como replicação. Ela é a mais comum de se encontrar nos sistemas hoje em dia e consiste em, a partir de um crescimento na demanda, criar novas máquinas virtuais, com a mesma configuração da MV original, para dividir a carga de trabalho entre todas elas. Este é o único método elástico que fornece um modelo de alocação baseado em conjuntos fixos, uma vez que é impossível alterar a configuração de uma MV [6].

Na Figura 3.3, vemos em (a) que há apenas uma máquina virtual em execução, se esta máquina ficar sobrecarregada com o aumento na demanda, a replicação acontece, e novas MVs,

idênticas a inicial, são alocadas para atender este crescimento na demanda (b). O trabalho, então, é dividido entre elas por um balanceador de carga.

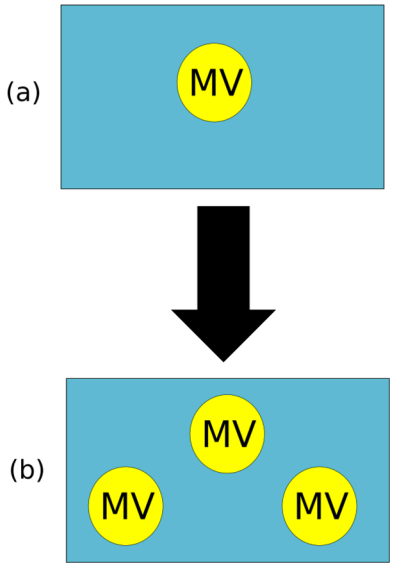

Figura 3.3: Exemplo de replicação de uma máquina virtual.

3.3. Vertical

A elasticidade vertical, também chamada de redimensionamento, engloba tanto alterações de configuração de recurso quanto a adaptação de aplicações [5]. No primeiro caso, é possível modificar a porcentagem da CPU, memória, disco ou largura de banda, destinada aquela aplicação. Já no segundo caso a aplicação reage a inclusão de novos recursos (CPU, memória, disco, etc.) e a aplicação pode criar novas *threads* ou novos processos para usufruir desta nova configuração da infraestrutura [5].

A Figura 3.4 mostra o exemplo de redimensionamento em uma máquina virtual executando (a). Caso esta máquina virtual fique sobrecarregada com o aumento na demanda, ela pode alocar mais recursos físicos para suprir esta necessidade (b), e se o oposto acontecer, e ela tenha recursos ociosos, ela pode liberá-los (c), assim estes recursos ficam disponíveis para alguma outra MV que esteja precisando e esteja no mesmo servidor físico.

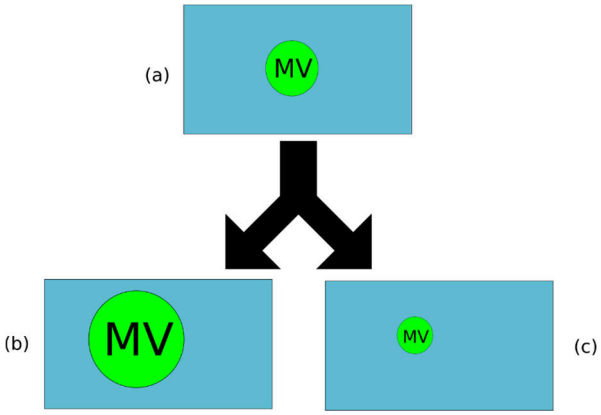

Figura 3.4: Exemplo de redimensionamento de uma máquina virtual.

3.4. Suporte à elasticidade

Atualmente, os principais hipervisores disponíveis oferecem suporte a elasticidade horizontal, permitindo que várias máquinas virtuais possam ser instanciadas, porém isto não ocorre com a elasticidade vertical. Na Tabela 3.1, adaptada de Galante [6] mostra as características da elasticidade vertical oferecida pelos principais hipervisores atuais: VMWare vSphere, KVM, Xen, e Microsoft Hiper-V, podendo-se concluir que elasticidade vertical ainda não é totalmente suportada.

Tabela 3.1: Principais hipervisores atuais e seus respectivos suportes à elasticidade.

Hipervisor	**Características**
Hiper-V	Memória (*ballooning*), discos (expansão e adição).
KVM	CPU (somente adição), memória (*ballooning*), disco (adição), interfaces de rede (adição).
vSphere	CPU (adição e remoção), memória (via *ballooning*), discos (expansão e adição), interfaces de rede (adição), e PCI express.
Xen	CPU (adição e remoção), memória (*ballooning*), discos (expansão e adição), interfaces de rede (adição).

A partir da Tabela 3.1, podemos ver que o hipervisor vSphere é o mais avançado em questão de elasticidade vertical, porém, como é um sistema proprietário, o seu uso em pesquisas acadêmicas torna-se inviável. Por este motivo optou-se, para a realização deste trabalho, utilizar o Xen, que possui suporte as funcionalidades que desejamos implementar (adição e remoção de CPU e memória), e é um projeto *Open Source*, o que facilita a busca por documentação. O Xen funciona com o sistema de Domínios (*domains*), o *Domain 0*, ou Dom0, é o domínio que inicia junto com o hipervisor, ele é uma máquina virtual com privilégios especiais que permitem

acesso direto ao hardware, controle de funções de entrada e saída e interação com outras máquinas virtuais, chamadas DomUs. Os *Unprivileged Domais*, ou DomUs, são máquinas virtuais independentes, executando seus próprios Sistemas Operacionais e aplicações, porém, diferente do Dom0, não tem nenhum acesso ao hardware ou funções de entrada e saída [26].

Logo acima do Hipervisor, há a camada de gerenciamento da nuvem que fornece uma ampla visibilidade e controle, ao mesmo tempo que permite que o usuário acesse, provisione e gerencie serviços de TI entre várias nuvens e plataformas [27].

A Tabela 3.2, adaptada de Galante [6], mostra que, assim como nos hipervisores, os provedores de nuvens implementam, principalmente, soluções elásticas horizontais, e deixam a desejar quando se trata de elasticidade vertical. Os poucos provedores que possuem esta funcionalidade, fazem isto de maneira parcial e/ou manual. Além disto a oferta de recursos destes provedores é feita, em maioria de maneira fixa, o que significa que as máquinas virtuais já vêm com quantidades pré-definidas de CPU, memória, rede, etc. que mesmo podendo contar com uma grande quantidade destas configurações, ela pode não atender a necessidade de todos os clientes.

Tabela 3.2: Algumas soluções elásticas e suas configurações.

Nuvem	Oferta de recursos	Elasticidade suportada
Provedores públicos		
Amazon	Fixa – 22 configurações	Horizontal
Rackspace	Fixa – 17 configurações	Horizontal
Azure	Fixa – 10 configurações	Horizontal
GoGrid	Fixa – 60 configurações	Horizontal e vertical (memória)
ProfitBricks	Flexível	Horizontal e vertical
CloudSigma	Flexível	Horizontal
Plataformas para nuvens privadas		
Eucalyptus	Fixa	Horizontal
OpenStack	Fixa	Horizontal
OpenNebula	Flexível	Horizontal

Foi pensando em solucionar estas falhas que este trabalho foi desenvolvido, aplicando elasticidade e automação para tentar se adequar ao máximo a necessidade do usuário. A metodologia de como estas ferramentas serão utilizadas para a realização deste trabalho estão no Capítulo 4.

Capítulo 4

Implementação de mecanismos automáticos de elasticidade

Como foi visto na Seção 3.4, as infraestruturas de nuvem atuais, oferecem algum suporte à elasticidade, porém este suporte é, na grande maioria, limitado a replicação (elasticidade horizontal) onde os recursos alocados são máquinas virtuais inteiras, oferecidas com valores fixos de recursos, como CPU e memória. Além disto, não há a possibilidade de migração entre os tipos de instância sem a necessidade de reinicialização da mesma [6].

Portanto, a elasticidade fornecida pelos atuais provedores é bastante limitada e a forma como os recursos são fornecidos pode não refletir a necessidade exata das aplicações, como prevê o modelo de computação em nuvem.

Este capítulo exibe uma solução proposta para resolver este problema e uma maneira de implementá-la. Estes tópicos estão nas Seções 4.1 e 4.2, respectivamente.

4.1 Solução proposta

O propósito deste trabalho é desenvolver a implementação de um mecanismo de elasticidade vertical e horizontal, que reaja de maneira automática ou manual, a partir de regras pré-definidas. Este sistema foi desenvolvido em Java, utilizando o hipervisor Xen e sistema operacional Linux.

Para implementar a elasticidade automática será utilizado um sistema de monitoramento que acompanhe a quantidade de recursos que cada máquina virtual está utilizando e também da máquina física, para verificar quanto recurso ainda há disponível.

A partir da análise dos dados gerados pelo monitor, o sistema consegue diagnosticar que uma determinada MV necessita de mais recursos e então fornecê-los, caso estejam disponíveis no hospedeiro físico, e também liberar recursos ociosos das MVs, de maneira automática.

O sistema foi implementado utilizando a abordagem reativa, onde o usuário configura previamente um conjunto de regras do tipo regra-condição-ação, que funcionam no esquema SE condição ENTÃO ação. Esta condição será a análise de um histórico de dados coletados

pelo monitor, que é comparado com algum limiar, (por exemplo, uso da CPU maior que 70%, uso da memória menor que 30%, etc.) e a ação poder ser o aumento ou diminuição dos recursos (alocar mais uma CPU, desalocar 1 GB de memória, adicionar outra MV, etc.).

O sistema foi dividido em dois subsistemas, que se comunicam através da rede, utilizando Sockets funcionando no esquema de Cliente-Servidor, ambos feitos em Java. O sistema do Cliente é executado em todas as máquinas virtuais a partir do momento da inicialização, e captura as leituras do monitor do estado atual da MV e as envia para o sistema Servidor. O Servidor recebe as informações das máquinas virtuais e analisa se é necessário realizar alguma alteração sobre ela. Caso seja necessário, o Servidor executa regras de alocação/liberação de recursos através do Xen e do Xen-tools[1] para tentar manter cada MV dentro das especificações inseridas pelo usuário. Este fluxo é exibido na Figura 4.1.

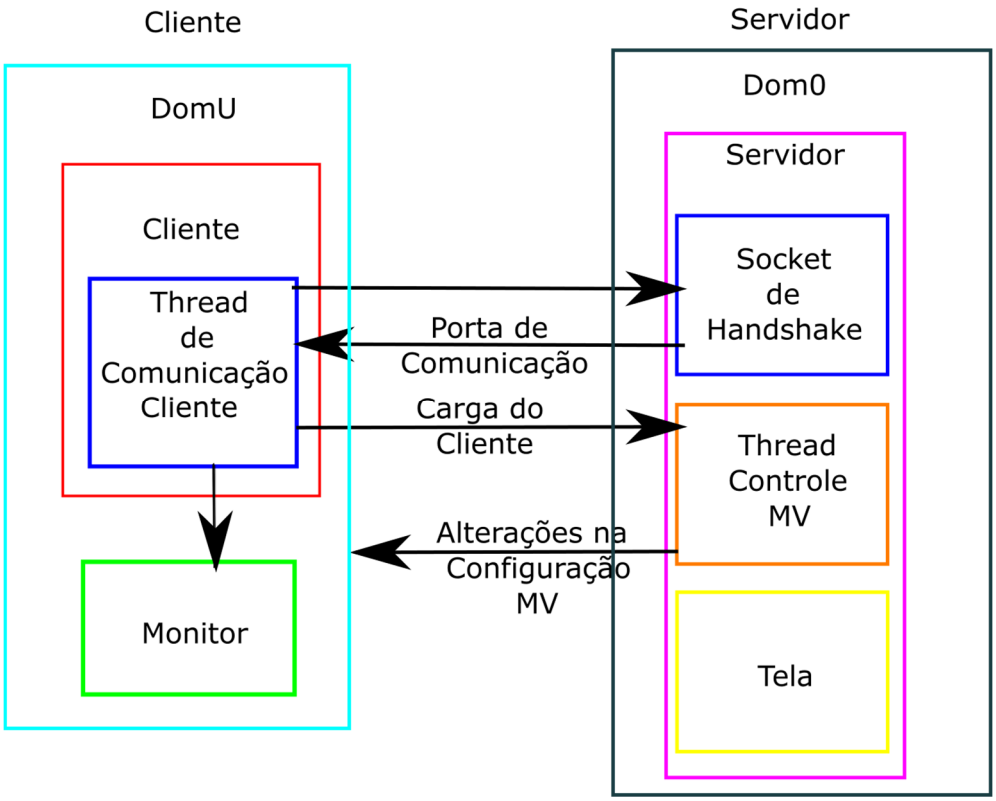

Figura 4.1: Mecanismo de funcionamento do sistema.

[1] Xen-tools é uma coleção de scripts em Perl que permitem ao Xen facilmente criar novos DomUs sobre um sistema Debian GNU/Linux [28].

4.2. Arquitetura do Sistema

Como descrito na Seção anterior, o sistema é dividido em duas partes, o Cliente, que é executado nos DomUs e coleta os dados do monitor para enviá-los para o Servidor, e Servidor, que recebe os dados e faz a análise afim de verificar se é necessário gerar uma alocação/remoção de recursos. A forma com que estes sistemas foram implementados está descrita nas subseções 4.2.1 e 4.2.2

4.2.1. Cliente

O sistema Cliente é iniciado assim que a máquina virtual termina o processo de *boot*, ele executa uma *thread* que se comunica com o sistema Servidor através de uma rede do tipo *bridge* configurada previamente. Uma vez que ele recebe a resposta do servidor, o Cliente passa para o modo de captura, em que ele executa um comando do monitor Dstat, obtendo os dados sobre a carga a cada 1 segundo e os envia para o Servidor através de um Socket. Caso a conexão se perca, ou o Servidor pare de executar, o Cliente volta ao estado inicial e tenta criar uma nova conexão com o Servidor até que ele responda.

Como dito anteriormente, o Dstat foi o Monitor usado para a coleta de informações. Com ele é possível obter uma visão geral dos recursos do sistema em tempo real de maneira fácil e clara. Ele mostra além de informações de CPU e memória, utilizadas neste trabalho, várias outras informações como rede, disco, entrada e saída, etc. [29].

Neste trabalho, como apenas será utilizado as informações de memória e CPU, somente estas informações serão capturadas. Para isto foi utilizado o comando "*dstat -c -m*" onde "-c" e "-m" significam que a saída conterá estatísticas de CPU e memória respectivamente. Uma demonstração de captura é ilustrada na Figura 4.2.

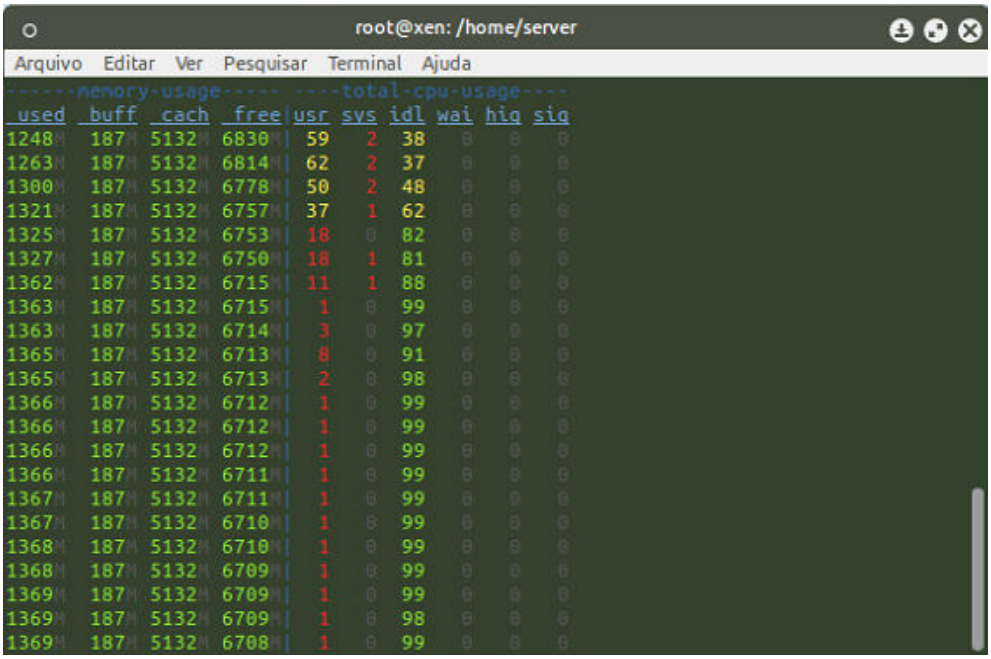

Figura 4.2: Estatísticas coletadas a partir do monitor Dstat.

4.2.2. Servidor

O servidor além de controlar os recursos das máquinas virtuais ele também faz o interfaceamento do sistema com o usuário, permitindo que ele monitore o estado das MVs, configure as métricas de alocação e remoção automática de recursos ou faça alterações manuais nestas propriedades.

Na parte da interface, é possível criar novas máquinas virtuais, alterar manualmente os recursos disponíveis para as MVs e também ativar o modo de elasticidade automático. Na parte de criação de uma máquina virtual, o usuário pode escolher o *hostname*, IP, quantidade de memória e VCPU que deseja alocar na inicialização e quantidade de memória e VCPU que a máquina virtual pode conseguir durante a sua execução. Quando o usuário clica no botão “Criar VM” o sistema passa as informações inseridas para o Xen-tools que cria um arquivo com estas configurações e faz a cópia de uma imagem já configurada, então o sistema executa o comando de criação de máquinas virtuais do Xen (*xl create diretório/para/arquivo/de/configuração.cfg*) fazendo com que a MV inicie. No modo manual, o usuário pode escolher quanto de memória e VCPU ele deseja aplicar para uma das MVs ativas, respeitando os limites escolhidos na hora da criação. Já o modo automático utiliza os dados recebidos do sistema Cliente e as configurações escolhidas pelo usuário para tentar manter os recursos disponíveis dentro de um limite julgado aceitável pelo usuário.

Para conseguir manter a carga da MV dentro destes limites o sistema utiliza uma *thread* que fica esperando receber requisições de algum cliente, quando isto acontece, uma nova *thread* é criada para ficar recebendo estas informações enquanto a *thread* anterior continua esperando por requisições de outros clientes. Quando os dados do cliente são recebidos, eles são tratados e armazenados em uma fila de tamanho limitado, quando esta fila enche, os dados mais antigos são descartados para a inclusão de dados mais novos. Esta fila é utilizada para análise do sistema, cada vez que um novo dado é inserido na lista, é calculada uma média simples das cinco informações mais recentes, esta média é comparada com as regras que o usuário inseriu na hora de ativar o controle automático, caso a nova informação satisfaça alguma das regras, uma ação de alocação ou remoção de recurso será executada com o valor informado em tela pelo usuário.

A Figura 4.3 mostra uma imagem da aba "Propriedades Automáticas" onde o usuário pode configurar as regras de elasticidade vertical ou horizontal. Na parte superior estão as opções da Elasticidade Horizontal, neste caso o usuário escolhe qual é a carga necessária para que uma nova máquina virtual seja alocada ou desalocada. A parte inferior funciona de maneira semelhante, porém em vez de alocar uma MV inteira, apenas recursos são alocados, de acordo com o informado.

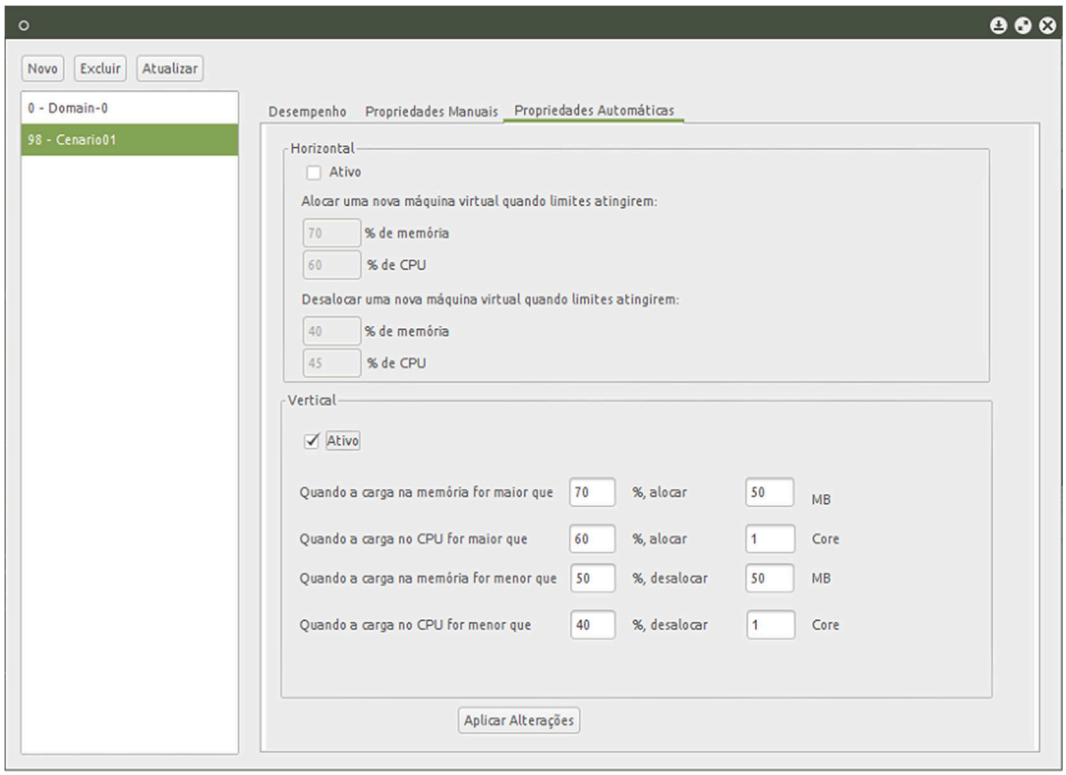

Figura 4.3: Exemplo de regras para a execução da elasticidade vertical.

Capítulo 5

Testes

Com base no sistema proposto e descrito no capítulo anterior, foram feitos testes para validar a solução. Este capítulo tem o objetivo de descrever os testes realizados tanto para a solução em elasticidade vertical quanto horizontal, para isto foram gerados cinco cenários de teste. Na Seção 5.1 são apresentados esses cinco cenários, juntamente com o comportamento do sistema utilizando os dois tipos de elasticidade. Na Seção 5.2 é apresentado alguns casos em que o sistema se comporta de maneira não convencional.

5.1. Cenários e comportamento do sistema

Os testes foram feitos em uma máquina Supermicro® modelo: X8DT3 com processador Intel® Xeon(R) CPU E5620 @ 2.40GHz × 16 e memória de 16 GB. Durante os testes, o Dom0 estava utilizando 8 GB de memória RAM e todos os 16 núcleos, porém, caso haja necessidade, os DomUs podem utilizar partes desta memória e os CPUs são compartilhados entre todos os domínios.

Os DomUs utilizados são padronizados, previamente configurados, com o sistema operacional Linux Ubuntu 15.04, inicialmente elas possuem 2 VCPU com o limite máximo de 4 VCPUs e 2048 MB de memória, com o limite máximo de 4096 MB, como exemplificado na tela de criação de novas máquinas virtuais do sistema na Figura 5.1. Para a realização dos testes foram utilizadas aplicações que simulam grandes cargas de memória e CPU e com isso acompanhar e validar apenas o comportamento de alocação e remoção de recursos ou máquinas virtuais de acordo com a carga alcançada e as configurações de elasticidade estipuladas. Foram feitos cinco cenários de teste e eles estão descritos nas subseções 5.1.1, 5.1.2, 5.1.3, 5.1.4 e 5.1.5.

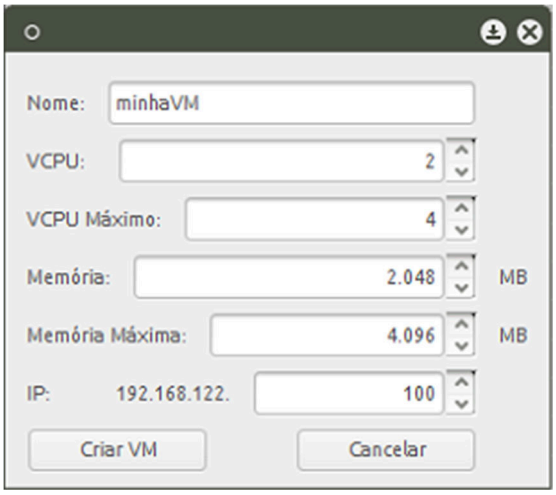

Figura 5.1: Modelo de configuração das MVs utilizado nos testes.

Os testes foram feitos com as propriedades automáticas preparadas para alocar recursos quando a média simples das últimas 5 capturas ultrapasse 70%, no caso de memória, e/ou 60% no caso de CPU. Quando a média das últimas 5 capturas for menor que 50%, de memória, e/ou 30% para CPU, ocorre uma liberação de recursos.

Ao iniciar um DomU, ele automaticamente começa a executar o sistema Cliente, porém aplicações usadas para simular demanda são executadas manualmente via SSH.

O sistema exibe um gráfico do tipo porcentagem de carga sobre os recursos disponíveis X tempo, assim foi possível exibir no mesmo gráfico as informações de CPU e memória, desta forma, mantendo a carga constante, quando um recurso é adicionado o gráfico desce, e quando algum recurso é removido, o gráfico sobe, causando a impressão de que ele esteja invertido. Porém, o aumento da carga também faz com que o gráfico suba podendo causar confusão em distinguir quando o sistema está desalocando recurso ou aumentando a demanda, para facilitar o entendimento foram adicionadas legendas com descrições de quando determinado evento acontece às imagens.

5.1.1. Cenário 1 – Teste de estabilização

Neste teste, uma máquina virtual foi criada, a partir do botão “Criar” do sistema, após o termino do *boot* a função de elasticidade automática foi habilitada para que apenas os recursos que ela realmente estava utilizando permanecessem alocados. Neste teste há apenas o sistema operacional e o Cliente em execução para assim tentar manter a carga constante durante todo o processo.

5.1.1.1. Abordagem Vertical

Quando uma MV acaba de ser criada, normalmente a carga dela é baixa, pois há poucos programas em execução, isto faz com que muitos recursos estejam ociosos, aplicando a elasticidade vertical, é possível notar que foi desalocada uma quantidade considerável de memória e a máquina virtual permaneceu apenas com o que realmente estava utilizando, dentro dos limites estipulados pelo usuário na guia de “Propriedades Automáticas” do sistema.

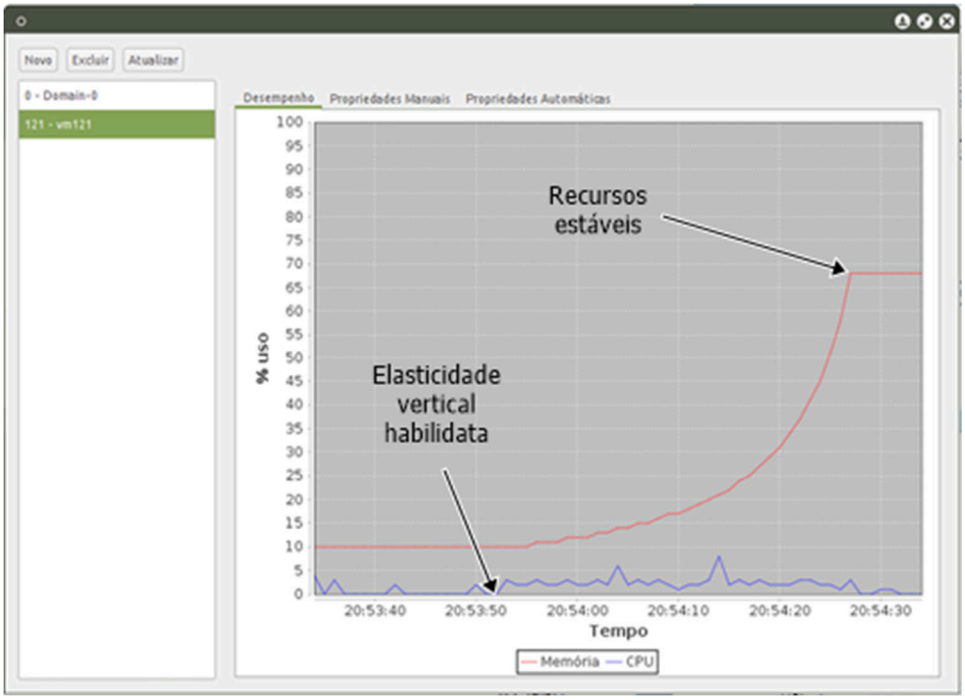

Figura 5.2: Comportamento Elástico Vertical no cenário 1.

5.1.1.2. Abordagem Horizontal

No cenário 1, a elasticidade horizontal apresentou-se ineficiente, como ela apenas aloca e desaloca máquinas virtuais inteiras, não é possível liberar nenhum recurso, mesmo que ocioso, pois só há uma máquina em execução.

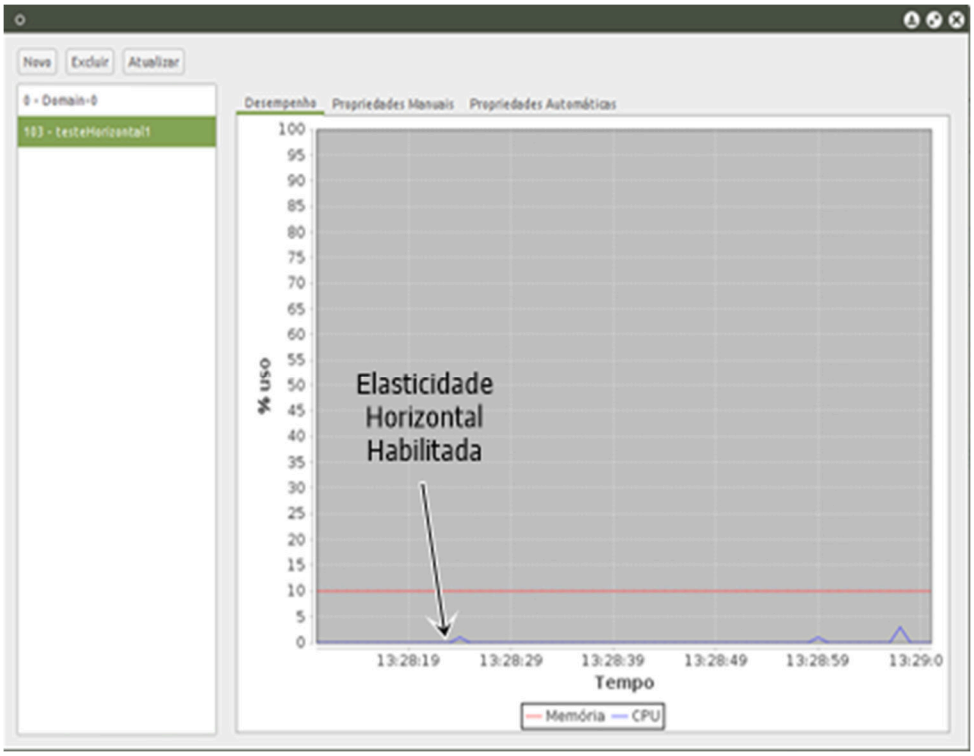

Figura 5.3: Cenário 1 aplicado a MV com elasticidade horizontal.

5.1.2. Cenário 2 – Teste de estresse de memória

A Partir da máquina com os recursos estáveis (Cenário 1) foi executado um programa auxiliar que gerava demanda de memória, ele foi configurado para consumir aproximadamente 14,5MB de memória a cada segundo durante 120 segundos, ocupando um total de, aproximadamente, 1.733 MB.

5.1.2.1. Abordagem Vertical

No cenário dois ocorrem várias alocações de memória, para suprir a necessidade gerada pelo aplicativo de teste, na Figura 5.4 é possível ver a linha do gráfico de memória subindo, apontando um aumento na carga, e vários picos correspondentes aos pontos de alocação deste recurso quando a porcentagem de uso ultrapassa o limite estipulado (70%). Após o termino do teste, o simulador de carga libera toda a memória que estava ocupada (Figura 5.5), fazendo com que a MV volte a ter bastante memória ociosa e o gráfico caia, o sistema então começa a desalocar memória, ocorrendo um comportamento parecido com o do cenário 1.

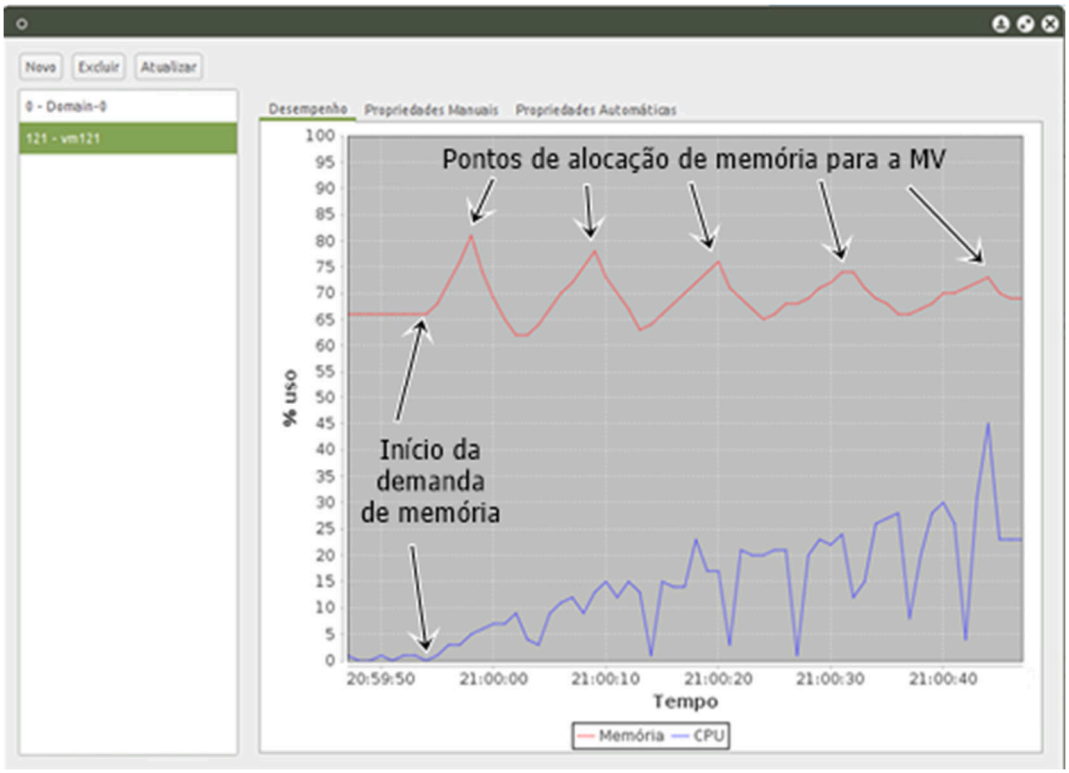

Figura 5.4: Inicio do comportamento elástico vertical no cenário 2.

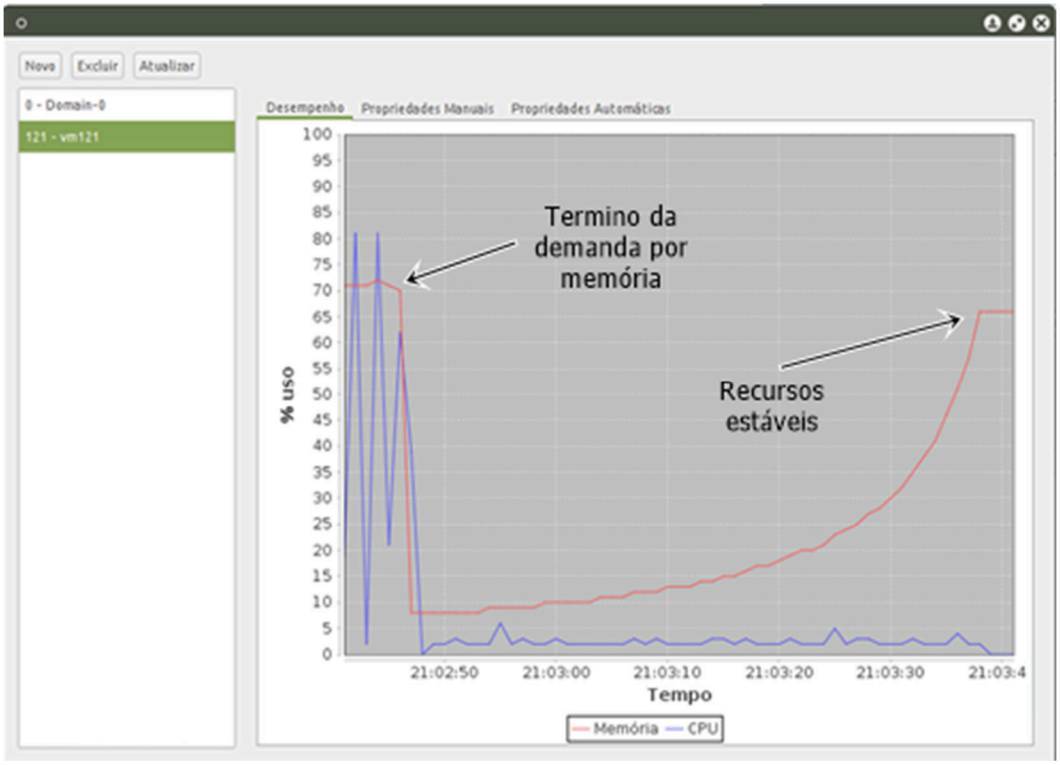

Figura 5.5: Final do processo no cenário 2 com elasticidade vertical.

5.1.2.2. Abordagem Horizontal

No cenário 2, a solução horizontal teve um bom início, e começou a alocação de uma nova MV quando ultrapassou o limite estipulado pelas configurações inseridas no sistema (Figura 5.7), porém com a demora para a criação da máquina virtual, o aplicativo simulador terminou a execução antes que a nova máquina começasse a funcionar (Figura 5.8), desta forma, quando estava pronta, não havia mais demanda, e assim ela foi desalocada (Figura 5.9). Fazendo com que todo o processamento gasto na criação desta nova MV fosse em vão.

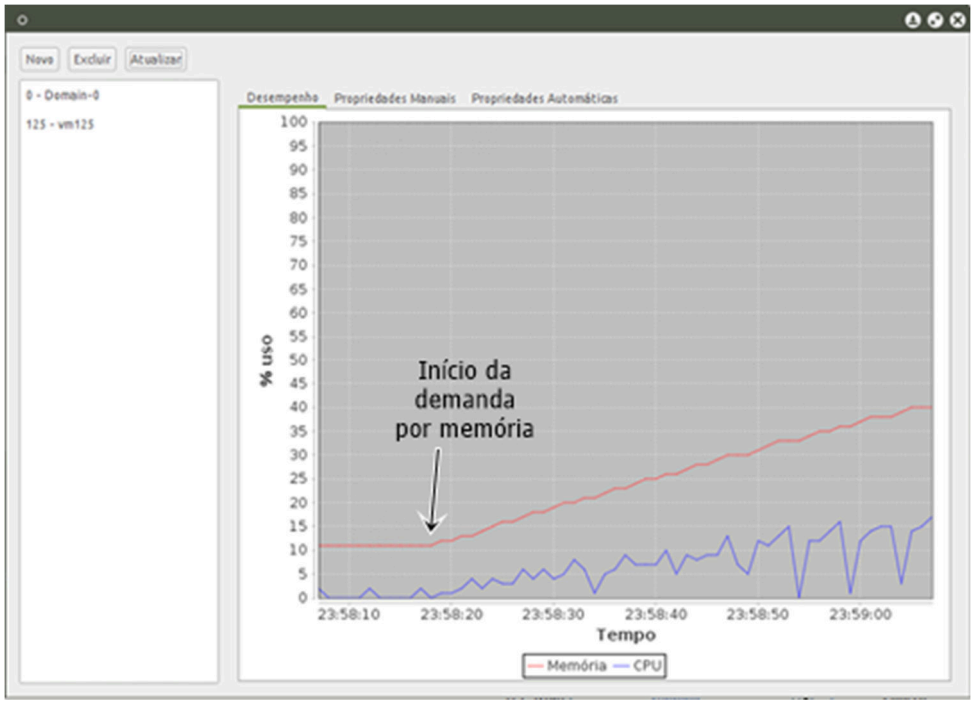

Figura 5.6: Inicio de alocação de memória.

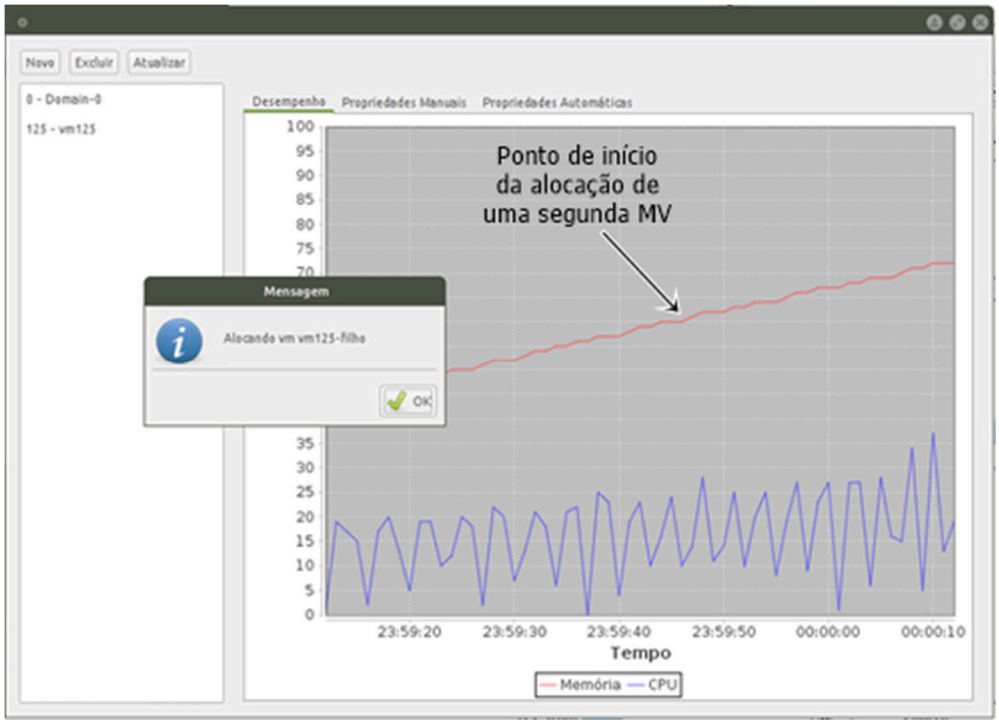

Figura 5.7: Inicio da alocação de oura MV.

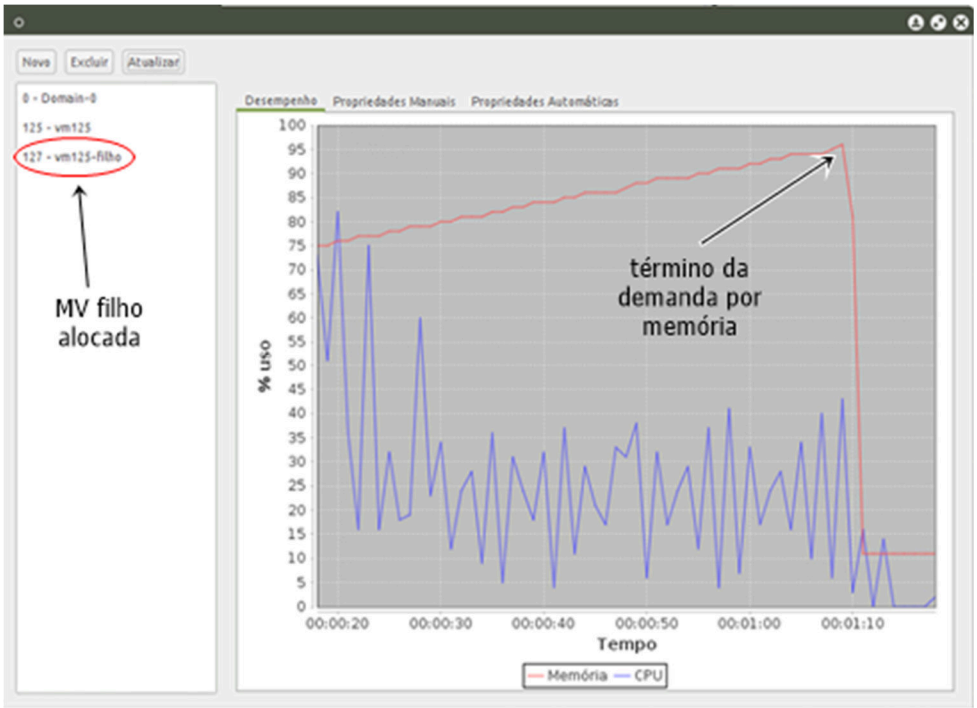

Figura 5.8: Término da alocação da segunda MV após o fim do processo.

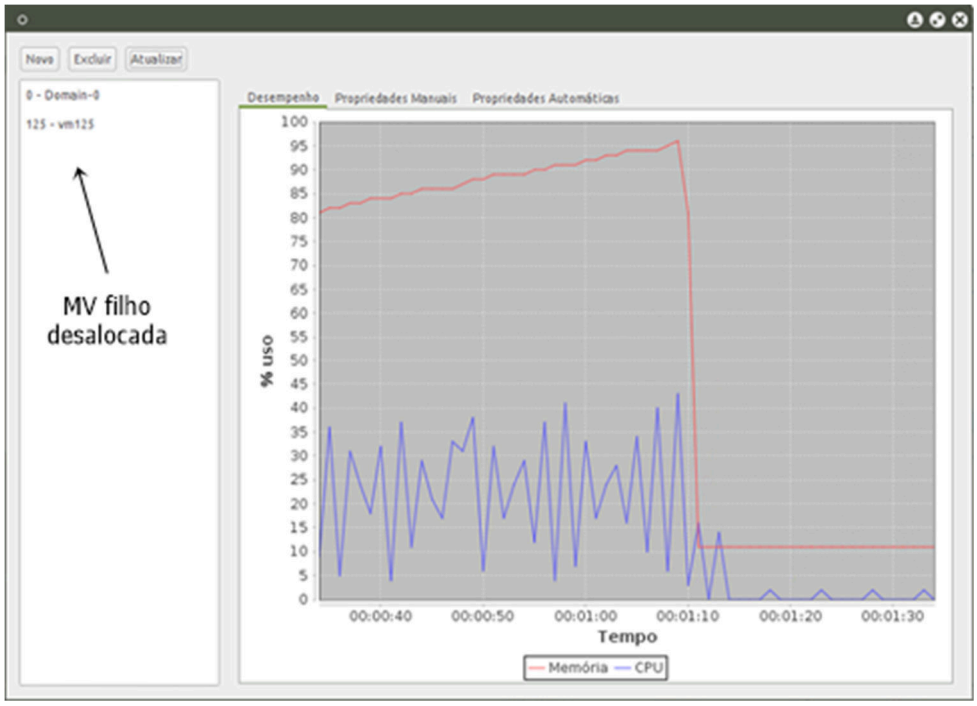

Figura 5.9: Desalocação da nova MV.

5.1.3. Cenário 3 – Teste de estresse de CPU

Neste teste foi executado um sistema auxiliar chamado “stress-ng”, com ele é possível simular uma demanda que ocupe determinada porcentagem de CPU além de quantas CPUS esta demanda ocupará. O comando utilizado foi o “*stress-ng -l 70 -c 2*”, este comando gera uma carga de 70% em 2 CPUs. O “stress-ng” não é 100% preciso na carga gerada, porém ele consegue mantê-la próximo do valor estipulado pelo usuário.

5.1.3.1. Abordagem Vertical

No início da execução, o sistema teve um pico de carga grande, pois como antes da execução do aplicativo auxiliar a carga era baixa, apenas uma VCPU estava alocada. Quando o teste começou o uso chegou a 100%, diminuindo com a alocação de novas VCPUs. Os recursos estabilizaram após alocar as 4 VCPUs disponíveis para a MV, assim, embora a carga ainda seja de 70% em duas VCPUs, há outras duas VCPUs sendo pouco utilizadas, mantendo o uso em aproximadamente 35% de uso total. Este processo pode ser visto na Figura 5.10.

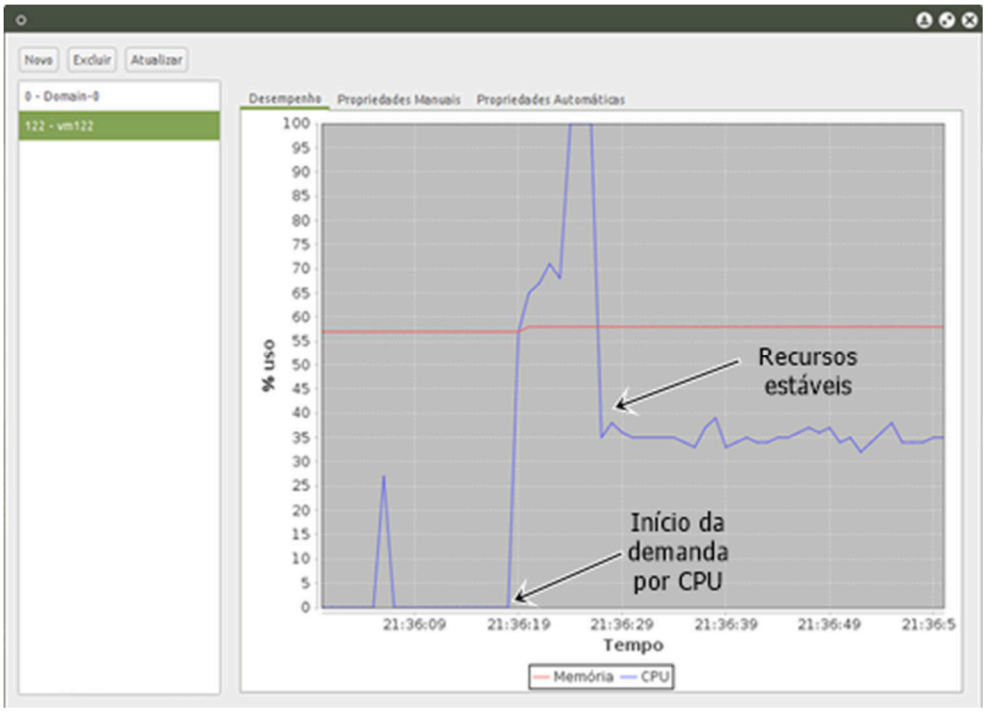

Figura 5.10: Comportamento elástico vertical no cenário 3.

5.1.3.2. Abordagem Horizontal

No cenário 3, a solução horizontal não apresentou o problema do cenário 2, já que o aplicativo que gerava a demanda não encerrou o processamento. A nova máquina virtual começou a ser alocada assim que a demanda iniciou (Figura 5.11), já que era maior que o limite estipulado para VCPUs (60%). Assim, a nova MV foi instanciada e ficou disponível para utilização, porém, como dito anteriormente, este trabalho está apenas avaliando a alocação e remoção de recursos das MVs, desta forma não foi utilizado um sistema que balanceasse a carga entre mais de uma máquina virtual, então a segunda MV continuou com muito recurso disponível (Figura 5.13), enquanto a primeira estava sobrecarregada (Figura 5.12).

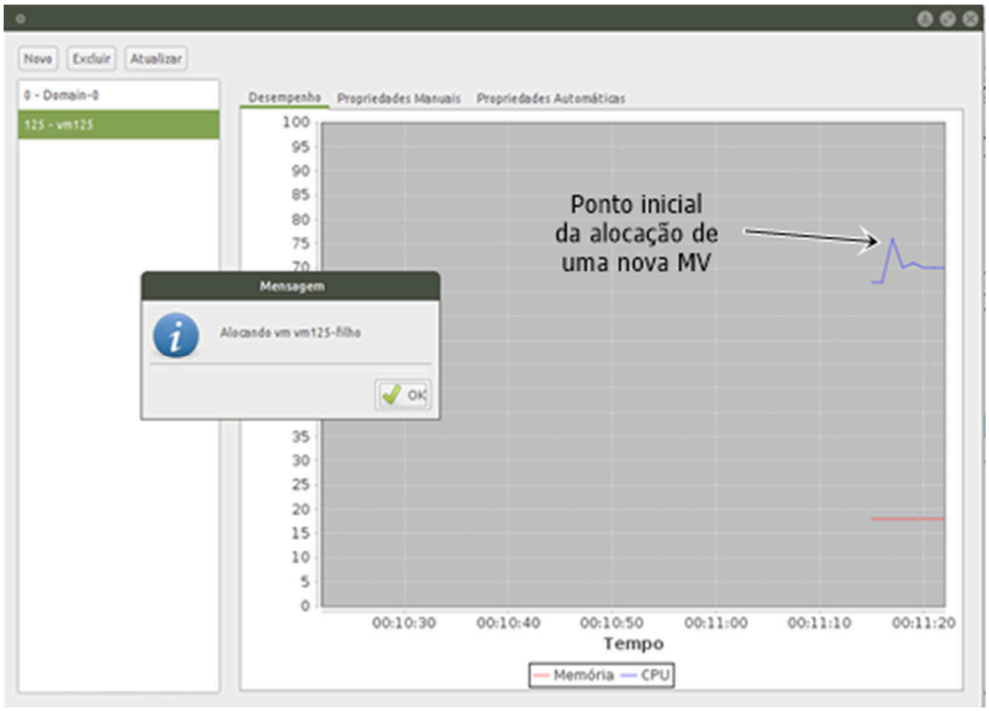

Figura 5.11: Início do processo e alocação da MV filho no cenário 3.

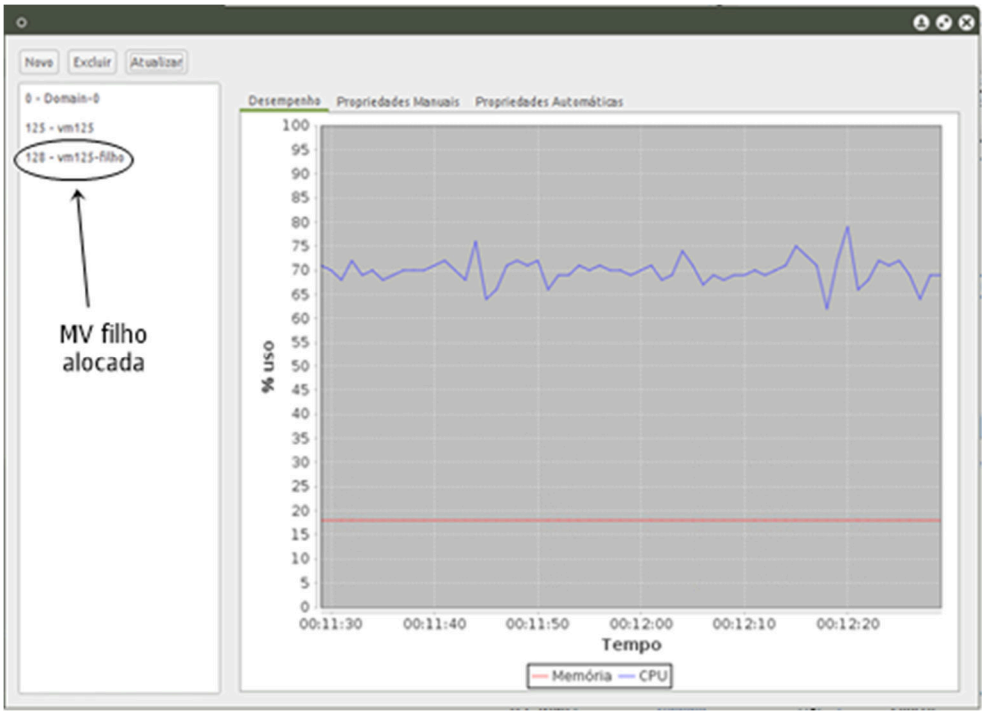

Figura 5.12: Carga na MV pai mesmo com uma MV filho criada.

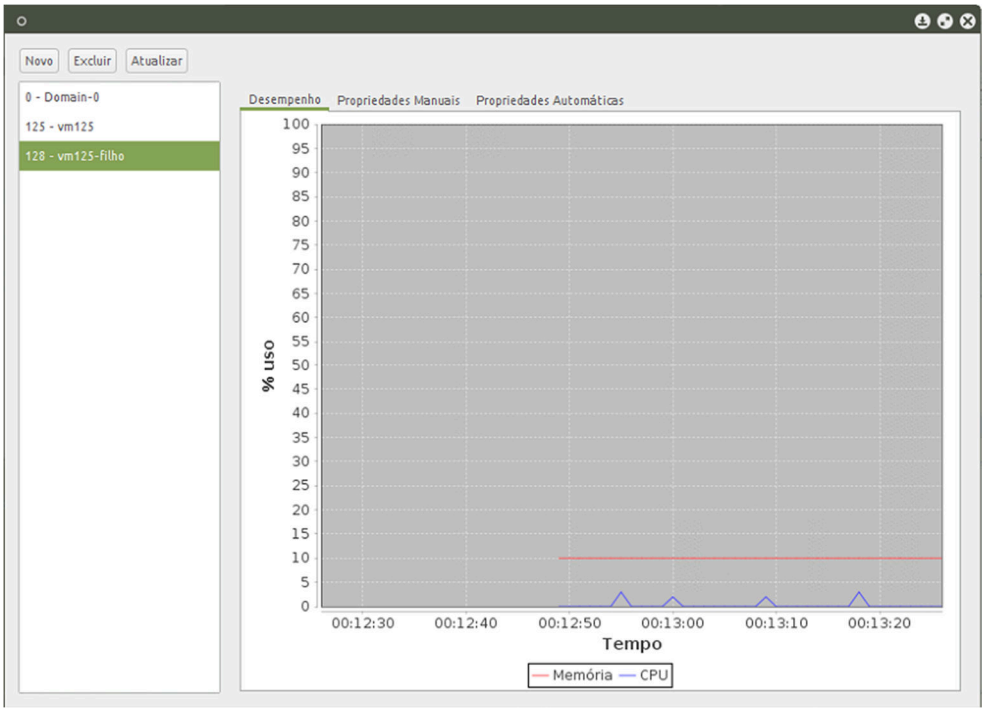

Figura 5.13: Carga na MV filho.

5.1.4. Cenário 4 – Teste com picos de estresse longos

Este teste executa ao mesmo tempo os cenários 2 e 3, porém com o tempo de duração de 2 minutos, então é feita uma pausa de aproximadamente 20 segundos e o teste é reiniciado, realizando este processo 3 vezes. Com este teste será possível ver como o sistema se comporta diante de mudanças de carga em sequência.

5.1.4.1. Abordagem Vertical

Utilizando a elasticidade vertical não houve grandes surpresas, o sistema se comportou de maneira semelhante aos Cenários 2 e 3, com a alocação rápida das quatro VCPUs disponíveis e com a alocação de memória sempre que a carga passa dos 70% gerando um gráfico com vários picos (Figura 5.14). Porém, agora é possível perceber uma variação maior na carga da CPU (Figura 5.15), provavelmente por haver os dois sistemas auxiliares executando ao mesmo tempo sobre o DomU além do próprio processamento do S.O. para gerenciar a constante alteração nos recursos e escalonar os processos em execução. Ao terminar o teste os aplicativos auxiliares são encerrados e a demanda cai, fazendo com que os recursos comecem a ser liberados (Figura 5.16) e a MV volta a utilizar apenas os recursos necessários para manter o sistema operacional em funcionamento.

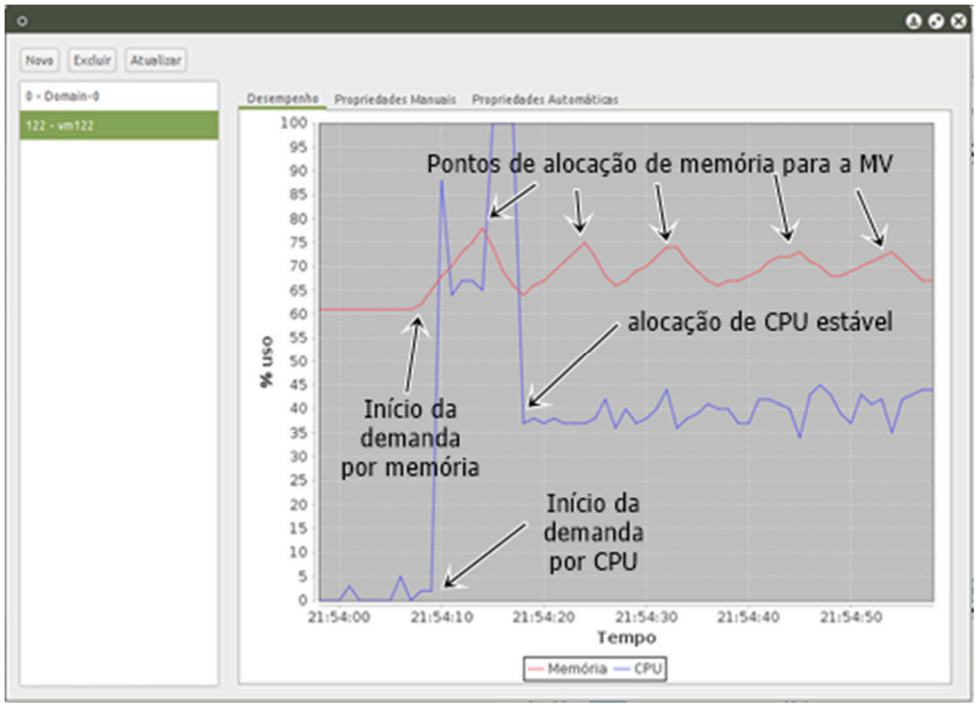

Figura 5.14: Início do comportamento elástico no cenário 4.

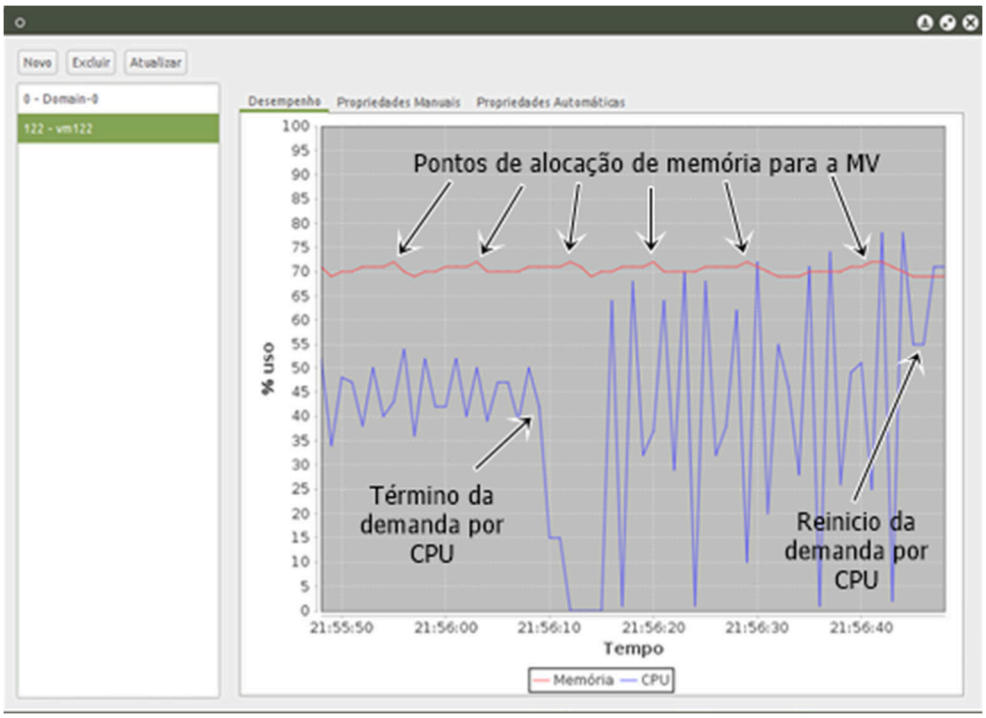

Figura 5.15: Término e recomeço da carga de CPU.

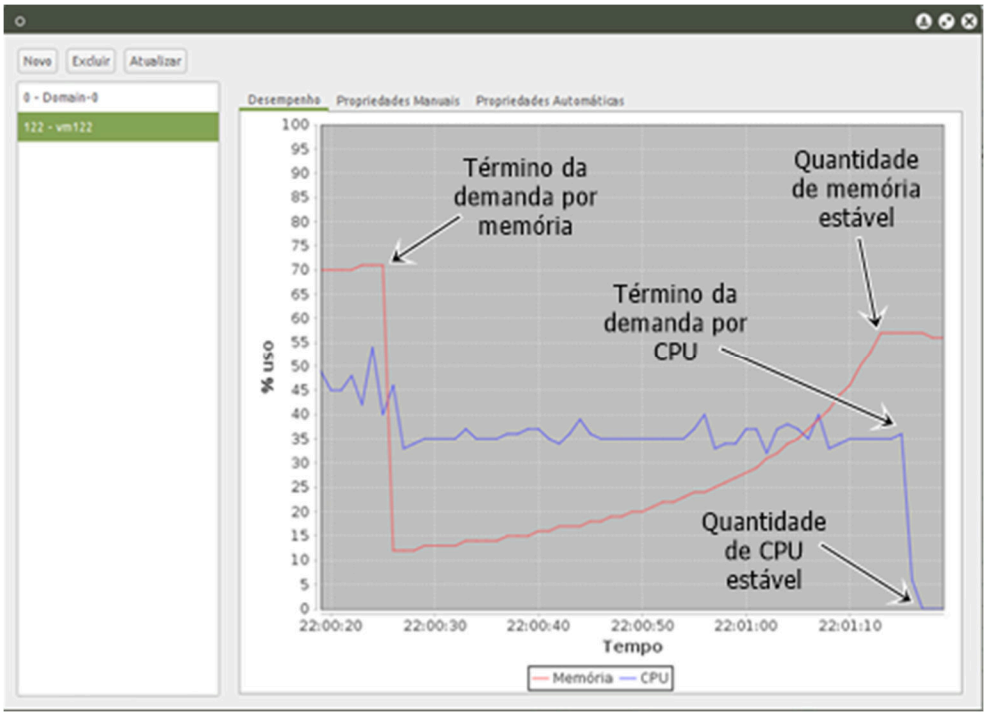

Figura 5.16: Fim da execução dos dois processos e estabilização dos recursos da MV.

5.1.4.2. Abordagem Horizontal

No cenário 4, a nova MV foi criada logo no início da execução (Figura 5.17), pois o uso da VCPU estava alto, e como a duração dos aplicativos de carga era de aproximadamente 2 minutos, e uma nova máquina virtual levam cerca de 1:20 minutos para iniciar, ela estava disponível para auxiliar na execução de tarefas. Quando os dois aplicativos geradores de carga estavam em pausa entre uma execução e outra, a nova MV foi desalocada (Figura 5.18), já que não era mais necessária naquele momento, quando a demanda de memória e VCPU voltaram a aumentar, uma nova máquina virtual foi criada (Figura 5.19). Por fim, quando o teste termina de executar, a demanda cai novamente e a segunda MV volta a ser excluída (Figura 5.20).

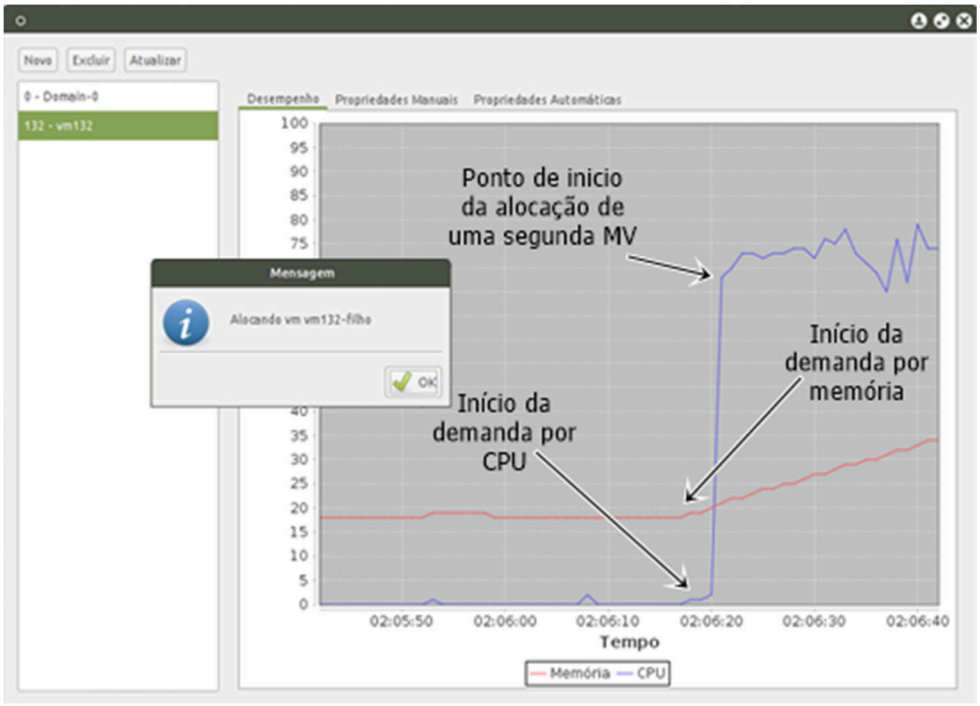

Figura 5.17: Inicio do cenário 4, alocação de uma nova MV.

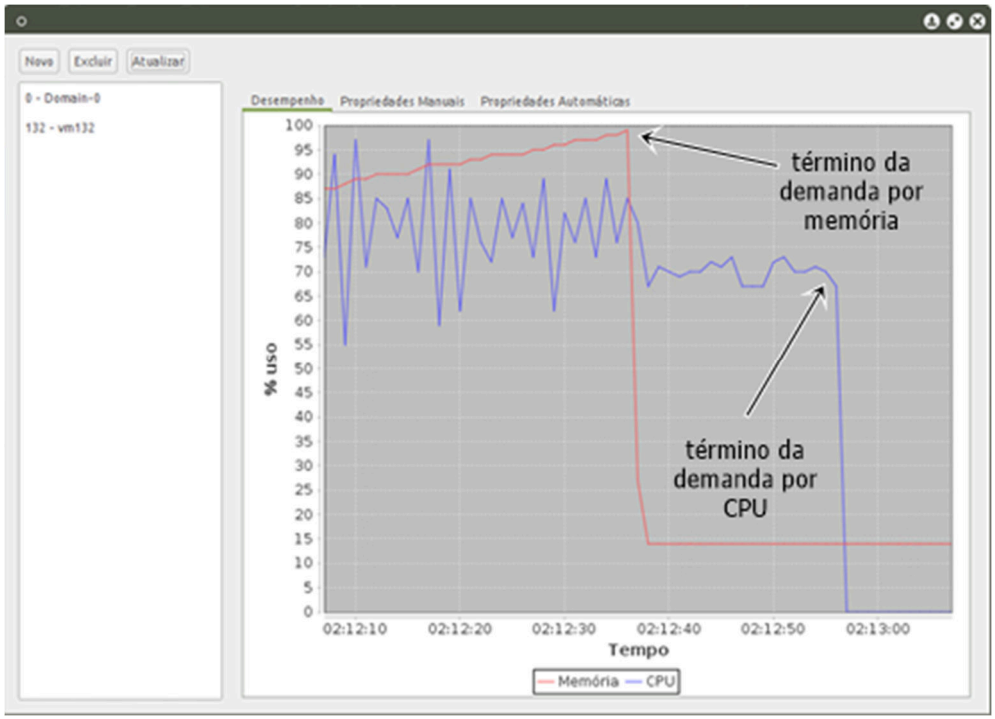

Figura 5.18: MV adicional excluída em um ponto que a carga diminui.

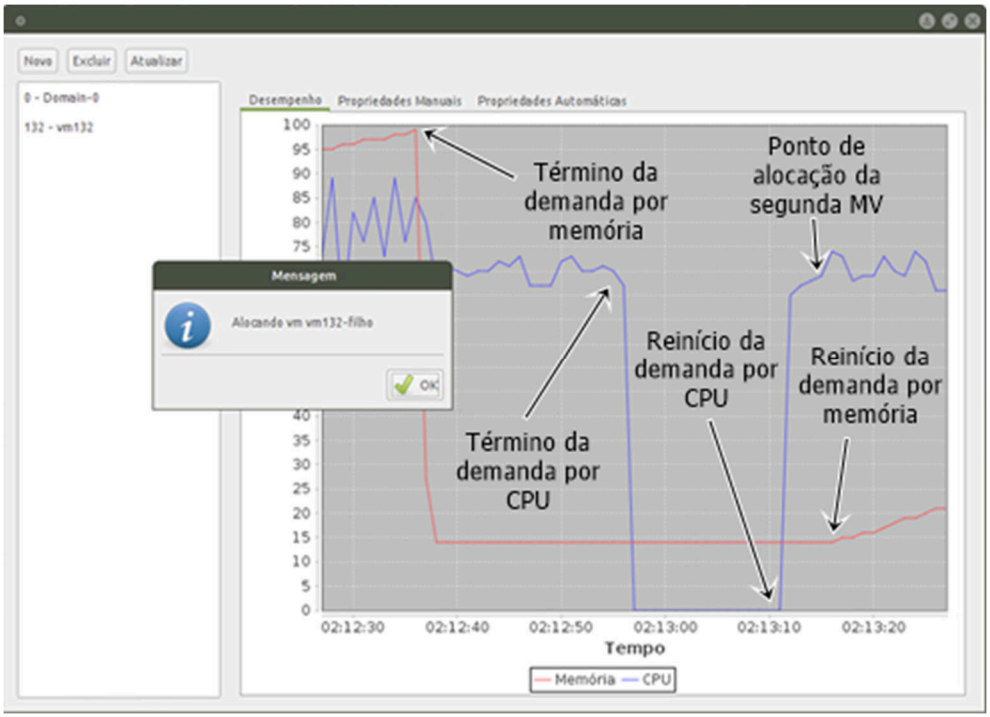

Figura 5.19: MV adicional volta a ser alocada quando a demanda aumenta.

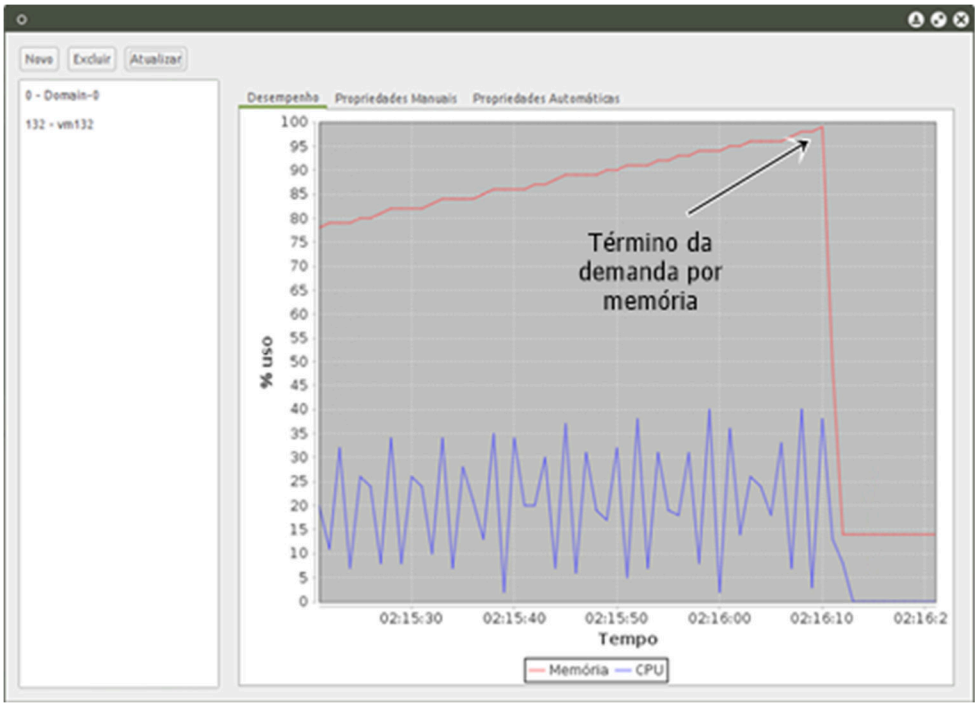

Figura 5.20: VM adicional volta a ser removida quando a carga diminui.

5.1.5. Cenário 5 – Teste com picos de estresse curtos

Este teste é semelhante ao anterior, porém com um tempo de duração dos processos menor e com alocações diferentes, foi utilizado uma carga de CPU de 80%, para 2 VCPUs, e para memória foi utilizado uma configuração no aplicativo auxiliar em que era ocupado 50MB de memória a cada segundo durante 20 segundos. O tempo mais curto de execução e a maior carga na memória faz com que o sistema tenha que se comportar de maneira rápida para conseguir atender bem a necessidade da MV. Os testes foram executados 3 vezes com duração de 20 segundos, com uma pausa de aproximadamente 10 segundos entre eles.

5.1.5.1. Abordagem Vertical

No cenário 5 também não houve muita diferença no comportamento do sistema, o gerenciamento de VCPUs seguiu o mesmo padrão, entretanto, como foi alterado as configurações do aplicativo gerador de demanda de memória e a quantidade de memória por segundo que ele ocupa aumentou, o formato do gráfico alterou, desta forma, em vez de se ter um crescimento lento onde era bem visível os pontos de alocação de memória, ele teve um grande aumento no início, até o sistema detectar que precisava alocar mais recursos para suprir a necessidade. As alocações fazem a porcentagem de uso baixar, até completar o ciclo de execução do aplicativo de demanda (Figura 5.21). Após finalizar os aplicativos auxiliares, o sistema possui bastante recurso ocioso e começa o processo de liberação, até que os aplicativos auxiliarem reiniciem a execução, fazendo com que volte a haver demanda e que novas alocações sejam realizadas. No final do processo o sistema consegue liberar todos os recursos ociosos (Figura 5.22).

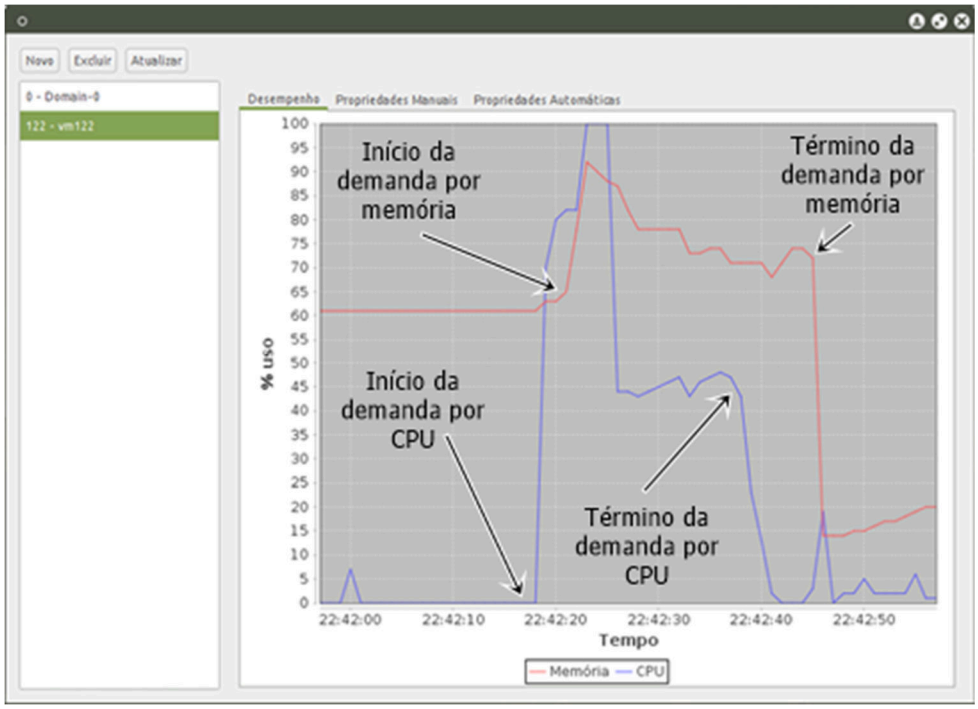

Figura 5.21: Início do processo do cenário 5.

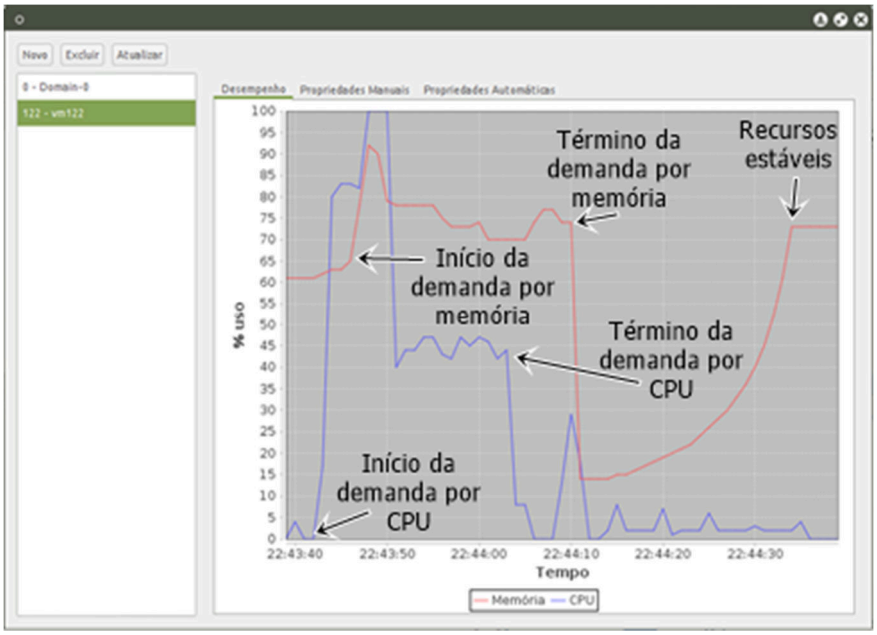

Figura 5.22: Final do cenário 5.

5.1.5.2. Abordagem Horizontal

No cenário 5, o desempenho da elasticidade horizontal não foi bom, a MV nova começou a ser alocada logo que o teste iniciou (Figura 5.23), porém, a demanda era rápida e passageira,

fazendo com que as três execuções dos testes ocorressem antes da máquina virtual nova começar a funcionar (Figura 5.25), que assim que entrou em funcionamento foi desalocada, pois já não havia mais demanda (Figura 5.26).

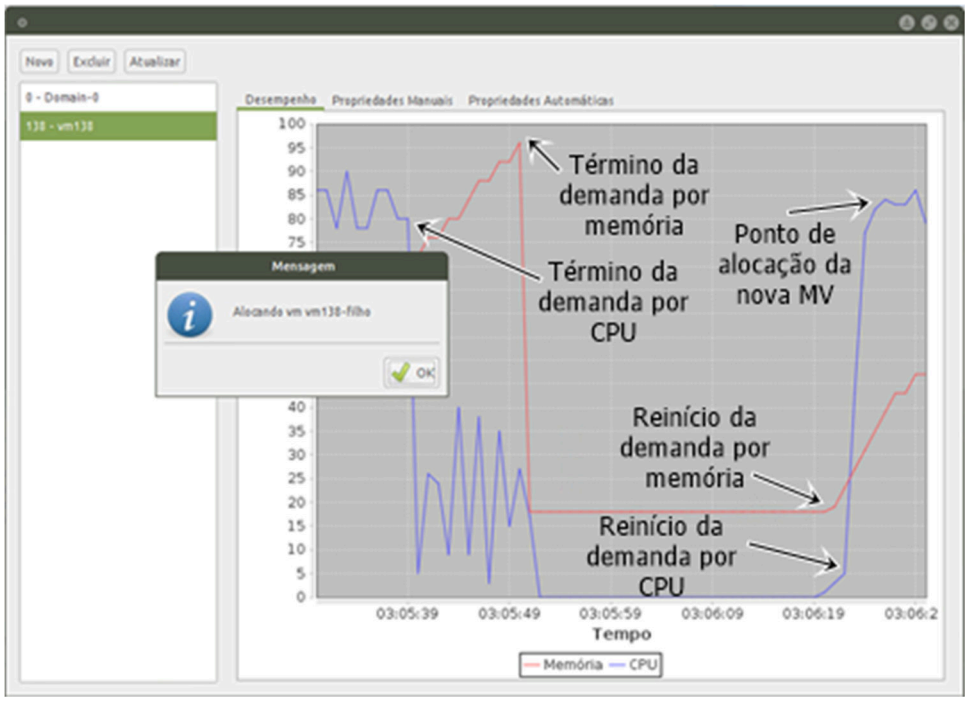

Figura 5.23: Inicio do cenário 5, alocação de outra MV.

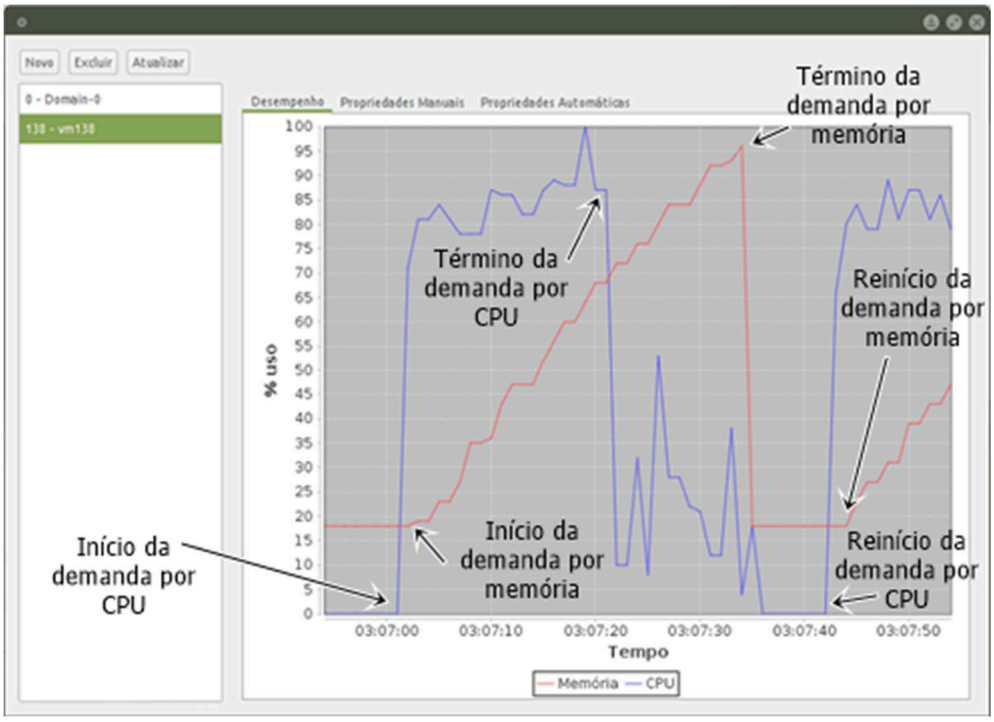

Figura 5.24: Demanda diminui e aumenta antes da MV nova estar pronta.

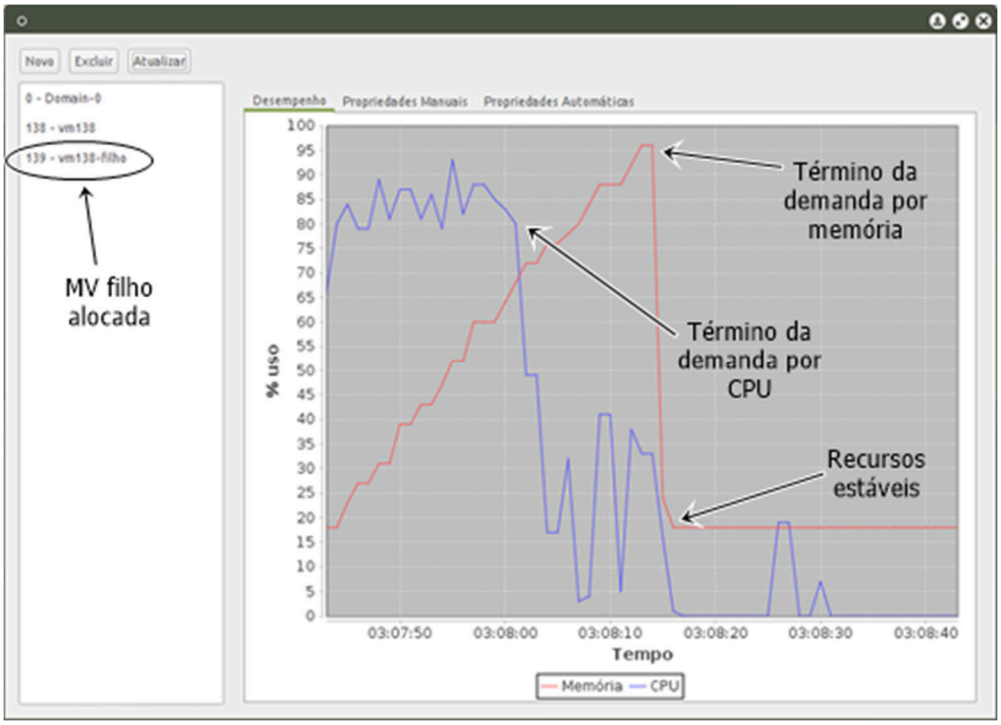

Figura 5.25: Após os processos terminarem de executar, a criação da nova MV termina.

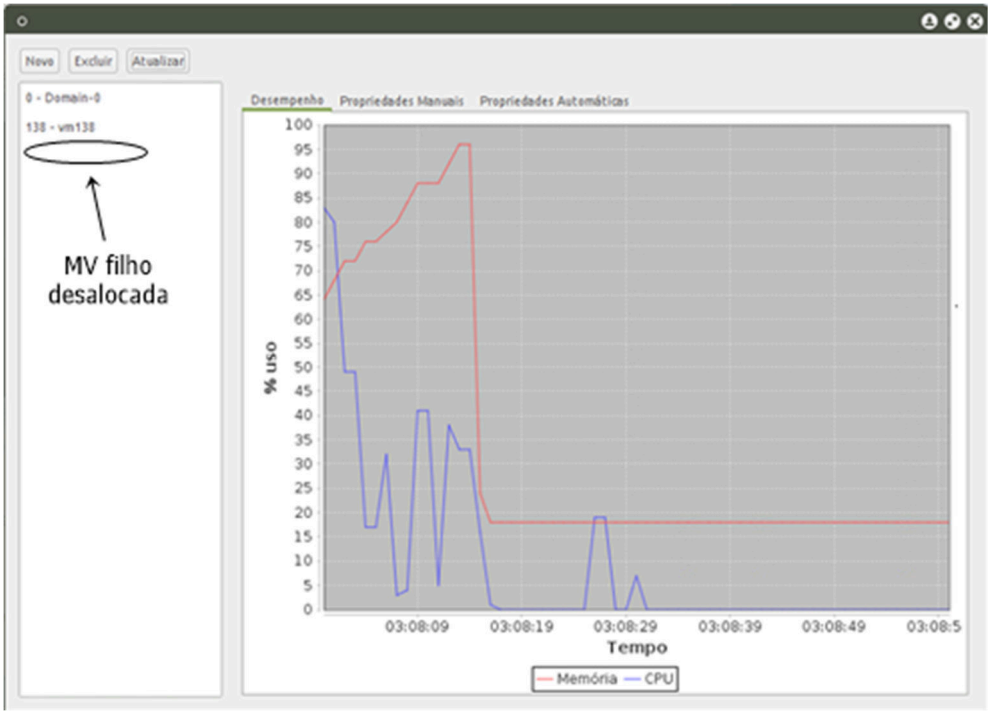

Figura 5.26: Em seguida, a MV é desalocada.

5.2. Casos Especiais

A partir destes testes e outros, também foi possível perceber que em determinados cenários, alguns casos especiais aconteciam, dependendo da configuração de alocação de recursos e dos sistemas utilizados para os testes. Uma má configuração de alocação pode gerar um mau desempenho da elasticidade, além de poder causar falhas tanto no sistema de monitoramento quanto no que está sendo executado sobre a máquina virtual, alguns destes casos serão apresentados a seguir.

5.2.1. Oscilação de recursos

Caso a quantidade de recursos alocados, por vez, seja muito grande, o sistema pode não conseguir estabilizá-los para a carga que está em execução, causando alocações e desalocações desnecessárias, na Figura 5.27 é possível ver um exemplo disto. Quando a MV necessita de mais memória, é realizada uma alocação deste recurso, porém se uma grande quantidade de memória for alocada de uma vez, a memória total para a máquina virtual pode ser muito maior do que ela necessita. Neste caso como há memória em excesso, uma rotina de desalocação é realizada, porém, novamente como a quantidade de memória desalocada é grande, o sistema volta a ter pouco recurso, gerando uma nova alocação, e assim sucessivamente, fazendo com que os recursos disponíveis estejam sempre oscilando.

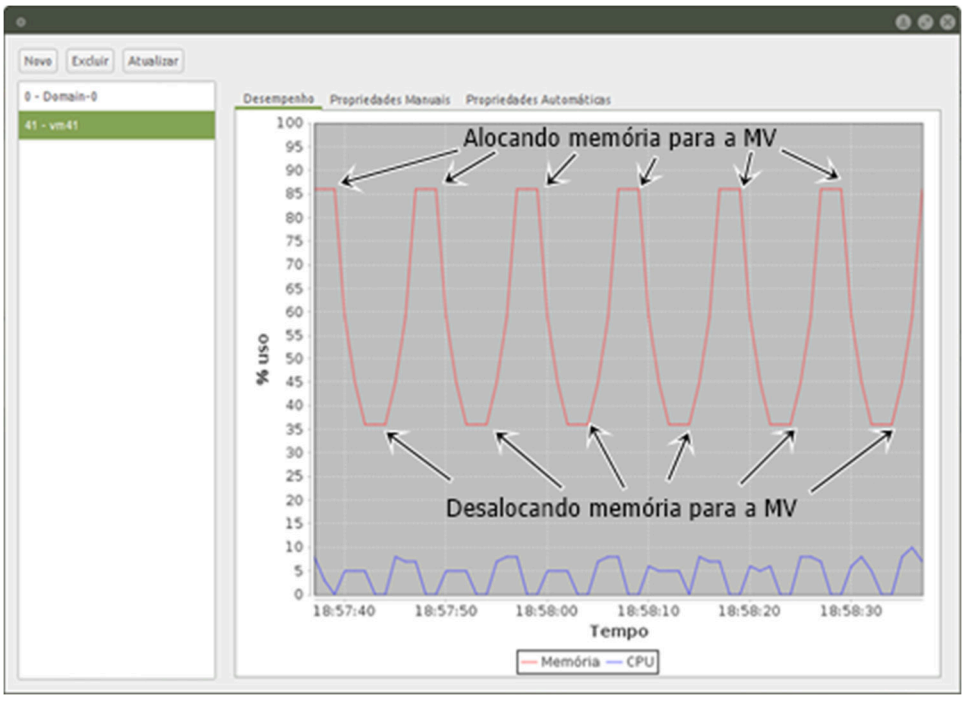

Figura 5.27: Exemplo de recursos com oscilação constante.

5.2.2. Falhas por excesso de demanda

Este caso pode ocorrer caso a quantidade de recursos alocados seja inferior ao crescimento da demanda, sendo assim, há uma necessidade de muito grande de recursos imediatos, porém a alocação não consegue acompanhar o ritmo da demanda, o que causa lentidão no sistema que está sendo executado sobre a máquina virtual e em casos mais graves, o software pode ser encerrado.

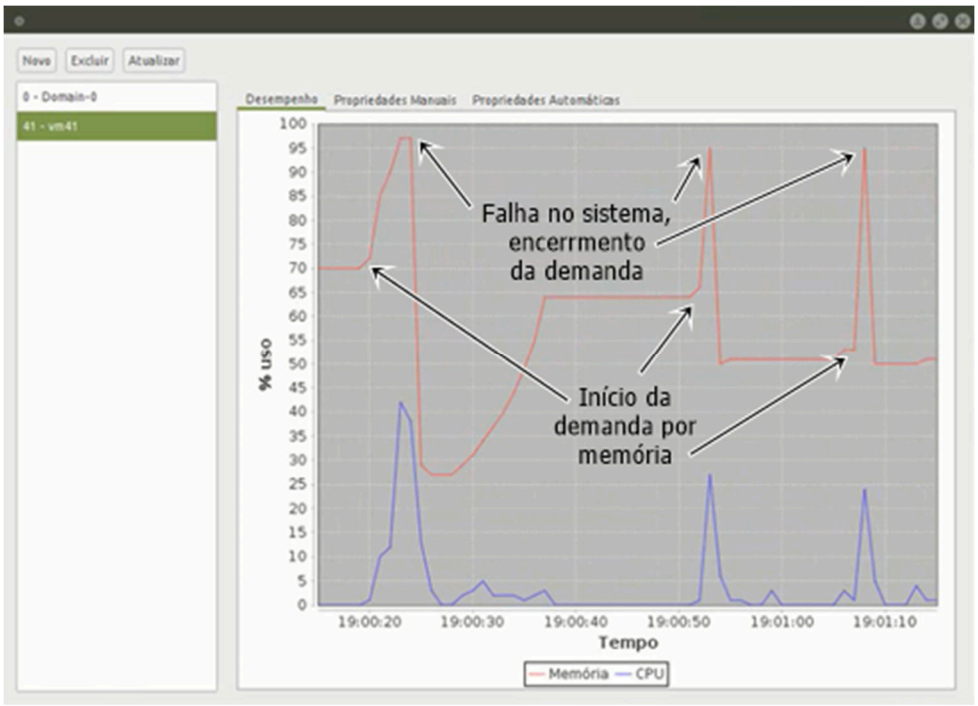

Figura 5.28: Exemplo de falhas por falta de recurso.

5.2.3. Utilizar o tipo inapropriado de elasticidade

A elasticidade, assim como a quantidade de recursos alocados por vez, deve ser compatível com o tipo de software que se beneficiará deste recurso. Como foi apresentado na Seção de testes 5.1 deste trabalho, não foi possível testar completamente o desempenho da elasticidade horizontal, pois os sistemas que foram utilizados para a realização dos testes não tinham suporte para dividir sua execução entre as duas máquinas virtuais disponíveis, então a carga sobre a primeira MV continuou alta, enquanto a segunda máquina virtual não estava sendo utilizada, desperdiçando todo o processo que o sistema teve de instanciá-la. O mesmo aconteceria caso um sistema que não fosse *multithread* executasse sobre um ambiente elástico vertical, várias outras CPUs poderiam ser alocadas para a máquina virtual, porém o sistema continuaria

utilizando apenas uma (Figura 5.29), fazendo com que a elasticidade nestes casos não seja aproveitada.

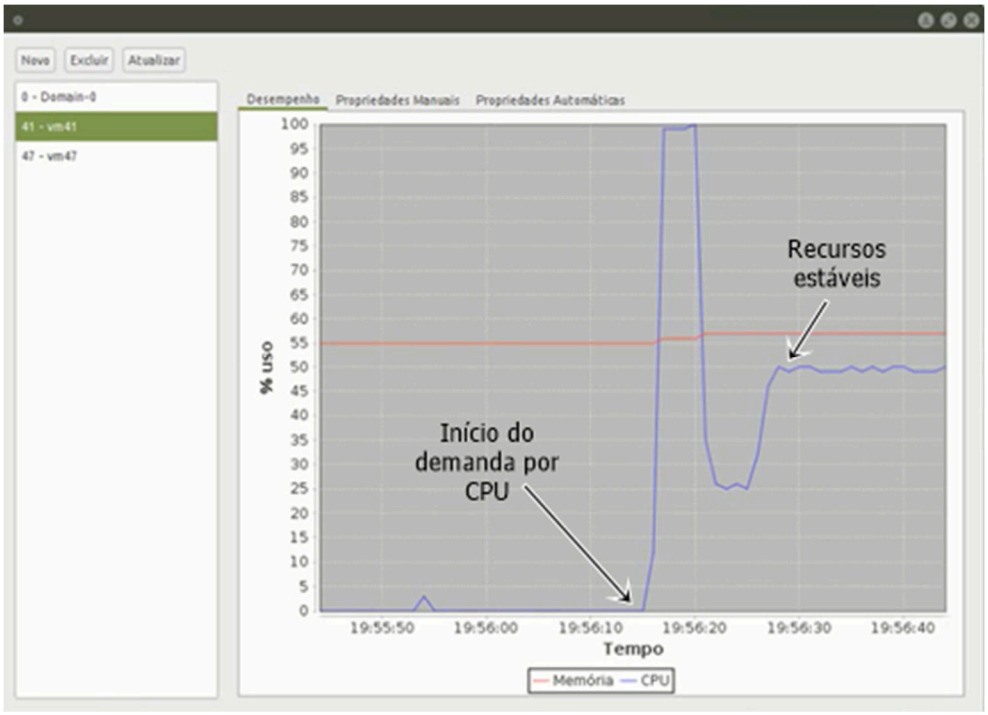

Figura 5.29: Exemplo de um sistema *single thread* em um ambiente com elasticidade vertical.

Capítulo 6
Conclusão

A partir dos dados coletados através dos testes descritos anteriormente, notou-se uma vantagem em utilizar a elasticidade vertical já que a alocação de recursos é rápida e pode ser feita em quantidades pequenas ou grande, conseguindo atender uma grande quantidade de casos e chegando mais perto da definição feita pelo NIST, além de conseguir ser bem aproveitado por sistemas *multithread* e não necessitar de outros sistemas auxiliares. Já a elasticidade horizontal não teve toda sua capacidade testada já que não foi utilizado nenhum tipo de balanceador de carga, nem aplicativo gerador de demanda que conseguisse agir de forma distribuída, limitando seus testes apenas a análise de alocação de novas MVs, neste caso ela se comportou de maneira inferior a elasticidade vertical por causa do tempo que a alocação de uma nova MV leva. Para demandas que duram bastante tempo, a demora na alocação não chega a ser um grande problema, porém, em testes em que era necessário que os recursos fossem adicionados rapidamente, ela teve um comportamento ruim, já que levava mais tempo para alocar uma nova MV do que durava a demanda.

Além do citado, outro problema na parte de gerenciamento das máquinas virtuais é estipular a quantidade de recursos que deve ser alocada, esta quantidade é muito dependente do comportamento dos sistemas que está executando na máquina virtual, tornando a utilização de valores fixos, como feitos neste trabalho, em aplicações reais, inviável. Caso os valores alocados sejam pequenos, é possível que não sejam suficientes para suprir as necessidades das aplicações, podendo causar lentidão e até falhas, e caso os valores alocados sejam muito grandes, pode ser que a máquina virtual possua muito recurso ocioso, além de comprometer a eficiência do mecanismo que controla a alocação/desalocação de um novo recurso, pois alocar grandes quantidades de recursos, pode gerar uma grande quantidade de recurso ocioso, causando então uma liberação de recursos, que se for em grande quantidade, também pode gerar uma nova alocação, e assim o sistema nunca se estabilizar.

6.1. Trabalhos futuros

- Utilizar meios estatísticos e outros parâmetros para tentar prever necessidades e alocar quantidades “mais precisas” de recursos;
- Utilizar meios de mesclar a solução vertical e horizontal;
- Testar o sistema proposto utilizando algum balanceador de carga;
- Expandir gerenciamento de elasticidade horizontal para controlar mais de uma máquina virtual;
- Expandir sistema para que possa controlar máquinas virtuais em mais de uma máquina física.

Referências Bibliográficas

[1] NATIONAL INSTITUTE OS STANDARDS AND TECHNOLOGY. **SP800-145**: The NIST Definition of Cloud Computing. Gaithersburg: Departamento de Comércio dos Eua, 2011. 6 p. Disponível em: <http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-145.pdf>. Acesso em: 08 mar. 2016.

[2] SANTOS, Uelinton; AMELOTTI, Luiz Augusto; VILLAR, Filipe. **White Paper:** Adoção de Computação em Nuvem e suas Motivações. 2012. Disponível em: <https://www.researchgate.net/publication/275519409_WhitePaper_-_Adocao_De_Computacao_Em_Nuvem_E_Suas_Motivacoes>. Acesso em: 08 mar. 2016.

[3] BADGER, Lee et al. **DRAFT Cloud Computing Synopsis and Recommendations.** 2011. Disponível em: <http://csrc.nist.gov/publications/drafts/800-146/Draft-NIST-SP800-146.pdf>. Acesso em: 08 mar. 2016.

[4] VAQUERO, Luis M.; RODERO-MERINO, Luis; BUYYA, Rajkumar. **Dynamically Scaling Applications in the Cloud.** 2011. Disponível em: <http://www.cloudbus.org/papers/ScalabilityInCloud2011.pdf>. Acesso em: 08 mar. 2016.

[5] RIGHI, Rodrigo da Rosa. Elasticidade em cloud computing: conceito, estado da arte e novos desafios. Revista Brasileira de Computação Aplicada, Passo Fundo, v. 5, n. 4, p.2-17, out. 2013. Disponível em: <http://www.upf.br/seer/index.php/rbca/article/view/3084/2370>. Acesso em: 08 mar. 2016.

[6] GALANTE, Guilherme. **Explorando a Elasticidade em Nível de Programação no Desenvolvimento e na Execução de Aplicações Científicas.** 2014. 131 f. Tese (Doutorado) – Pós-Graduação em Informática, Universidade Federal do Paraná, Curitiba, 2014.

[7] AMARAL, Fábio Eduardo. **O que é Virtualização?:** Uma visão geral sobre a Virtualização, tendência que revolucionou o mundo da TI. 2009. Disponível em: <http://www.tecmundo.com.br/web/1624-o-que-e-virtualizacao-.htm>. Acesso em: 08 mar. 2016.

[8] VAQUERO, Luis M. et al. **A Break in the Clouds: Towards a Cloud Definition.** 2009. Disponível em: < http://ccr.sigcomm.org/online/files/p50-v39n1l-vaqueroA.pdf >. Acesso em: 08 mar. 2016.

[9] P. Mell e T. Grance. Draft the nist definition of cloud computing. Nist Special Publication, 145(6):7, 2009.

[10] ROUSE, Margaret. **Public cloud.** 2009. Disponível em: <http://searchcloudcomputing. techtarget.com/definition/public-cloud>. Acesso em: 08 mar. 2016.

[11] MARTINS, Rômulo. **Pública, Privada, Comunidade ou Híbrida? Conheça os modelos de Cloud Computing**. 23 out. 2013. Disponível em: <http://www.qinetwork.com.br/publica-privada-comunidade-ou-hibrida-conheca-os-modelos-de-cloud-computing/>. Acesso em: 08 mar. 2016.

[12] FERREIRA, Anderson Vinícius Alves; BARROS, Davi Sabino; ALBUQUERQUE, Rafael Bezerra. **Serviços em Nuvem I:** Oportunidade para Operadoras. 2013. Disponível em: <http://www.teleco.com.br/tutoriais/tutorialservnuvopers1/pagina_3.asp>. Acesso em: 08 mar. 2016.

[13] MICROSOFT CORPORATION (Brasil). **Introdução ao Office.com.** 2016. Disponível em: < https://support.office.com/pt-br/article/Introdu%C3%A7%C3%A3o-ao-Office-com-91a4ec74-67fe-4a84-a268-f6bdf3da1804?CorrelationId=9f272038-6e3a-425a-882b-d42c646a7d55&ui=pt-BR&rs=pt-BR&ad=BR >. Acesso em: 08 mar. 2016.

[14] GOOGLE. **Google Drive.** 2016. Disponível em: <https://www.google.com/drive/>. Acesso em: 08 mar. 2016.

[15] SOUSA, Flávio R. C.; MOREIRA, Leonardo O.; MACHADO, Javam C.. **Computação em Nuvem: Conceitos, Tecnologias, Aplicações e Desafios.** 2010. Disponível em: <http://www.lia.ufc.br/~flavio/papers/ercemapi2009.pdf>. Acesso em: 08 mar. 2016.

[16] GOOGLE. **The App Engine Environments** 2016. Disponível em: <https://cloud.google.com/appengine/docs/the-appengine-environments>. Acesso em: 08 mar. 2016.

[17] MICROSOFT CORPORATION. **What is Azure?** 2016. Disponível em: <https://azure.microsoft.com/en-us/overview/what-is-azure/>. Acesso em: 08 mar. 2016.

[18] OWENS, Dustin. **Securing Elasticity in the Cloud.** 2010. Disponível em: <http://dl.acm.org/ft_gateway.cfm?id=1794516&ftid=320487&dwn=1>. Acesso em: 08 mar. 2016.

[19] HERBST, Nikolas Roman; KOUNEV, Samuel; REUSSNER, Ralf. **Elasticity in Cloud Computing: What It Is, and What It Is Not**. In: INTERNATIONAL CONFERENCE ON AUTONOMIC COMPUTING, 10., 2013, San Jose. **Proceedings.** San Jose: Usenix, 2013. p. 34 - 38. Disponível em: <http://0b4af6cdc2f0c5998459-c0245c5c937c5dedcca3f1764ecc9b2f.r43.cf2.rackcdn.com/11719-icac13_herbst.pdf>. Acesso em: 08 mar. 2016.

[20] ARMBRUST, Michael et al. **A View of Cloud Computing**. **Communications Of The Acm,** New York, v. 53, n. 4, p.50-58, 4 abr. 2010. Disponível em: <http://dl.acm.org/citation.cfm?id=1721672>. Acesso em: 08 mar. 2016.

[21] GALANTE, Guilherme; BONA, Luis Carlos E. de. **A Survey on Cloud Computing Elasticity**. In: IEEE/ACM INTERNATIONAL CONFERENCE ON UTILITY AND CLOUD COMPUTING, 5., 2012, Chicago. **Proceedings.** Chicago: Conference Publishing Services, 2012. p. 263 - 270. Disponível em: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6424959>. Acesso em: 08 mar. 2016.

[22] MARSHALL, Paul; KEAHEY, Kate; FREEMAN, Tim. **Elastic Site:** Using Clouds to Elastically Extend Site Resources. 2010. Disponível em: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5493493&tag=1>. Acesso em: 08 mar. 2016.

[23] CALHEIROS, Rodrigo N. et al. **The Aneka platform and QoS-driven resource provisioning for elastic applications on hybrid Clouds.** 2011. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0167739X11001397>. Acesso em: 08 mar. 2016.

[24] RODRIGUES, Juciely de Mesquita; SILVA, Rodrigo Ronner Tertulino da; LEITE, Jessica Neiva de Figueiredo. **Virtualização e Seus Benefícios para Empresas com Hyper-v; um Estudo de Caso na Indústria de Tempero Regina Ltda.** 2011. Disponível em: <http://www.aedb.br/seget/arquivos/artigos11/55714687.pdf >. Acesso em: 08 mar. 2016.

[25] REDAÇÃO DA COMPUTERWORLD. **Computação em nuvem pode gerar economia anual de US$ 12,3 bi com.** 2011. Disponível em: <http://idgnow.com.br/ti-corporativa/2011/07/22/computacao-em-nuvem-pode-gerar-economia-anual-de-us-12-3-bi-com-energia/>. Acesso em: 08 mar. 2016.

[26] XEN.ORG. **Xen Project Software Overview.** 2015. Disponível em: <http://wiki.xenproject.org/wiki/Xen_Project_Software_Overview>. Acesso em: 08 mar. 2016.

[27] VMWARE. **Soluções de gerenciamento em nuvem e virtualização.** 2016. Disponível em: <https://www.vmware.com/br/virtualization/virtualization-management/overview.html>. Acesso em: 08 mar. 2016.

[28] KEMP, Steve; BECKERT, Axel. **Xen Tools.** Disponível em: <http://xen-tools.org/software/xen-tools/>. Acesso em: 08 mar. 2016.

[29] DAG WIEËRS. **Dstat:** Versatile resource statistics tool. 2012. Disponível em: <http://dag.wiee.rs/home-made/dstat/>. Acesso em: 26 jan. 2016.