MINISTÉRIO DA DEFESA
EXÉRCITO BRASILEIRO
DEPARTAMENTO DE CIÊNCIA E TECNOLOGIA
INSTITUTO MILITAR DE ENGENHARIA
CURSO DE MESTRADO EM SISTEMAS E COMPUTAÇÃO

Maj EDGARD HONORATO CARDOSO BERNARDO

ARQUITETURA PARA SUPORTAR SOBRECARGAS MOMENTÂNEAS EM AMBIENTE DE COMPUTAÇÃO EM NUVEM

Rio de Janeiro
2014

INSTITUTO MILITAR DE ENGENHARIA

Maj EDGARD HONORATO CARDOSO BERNARDO

ARQUITETURA PARA SUPORTAR SOBRECARGAS MOMENTÂNEAS EM AMBIENTE DE COMPUTAÇÃO EM NUVEM

Dissertação de Mestrado apresentada ao Curso de Mestrado em Sistemas e Computação do Instituto Militar de Engenharia, como requisito parcial para obtenção do título de Mestre em Ciências em Sistemas e Computação.

Orientadores:
Prof Wallace Anacleto Pinheiro - D.Sc.
Profª Raquel Coelho Gomes Pinto - D.Sc.

Rio de Janeiro
2014

C2014

Praça General Tibúrcio, 80 – Praia Vermelha

Rio de Janeiro – RJ CEP: 22290-270

INSTITUTO MILITAR DE ENGENHARIA

004.22 Bernardo, Edgard Honorato Cardoso
B518a

Arquitetura para suportar sobrecargas momentâneas em ambiente de computação em nuvem / Edgard Honorato Cardoso Bernardo. Orientado por Wallace Anacleto Pinheiro e Raquel Coelho Gomes Pinto – Rio de Janeiro: Instituto Militar de Engenharia, 2014.

152p. : il

Dissertação (mestrado) – Instituto Militar de Engenharia – Rio de Janeiro, 2014.

1. Curso de Sistemas e Computação – teses e dissertações. 2. Arquitetura de Computadores. 2. Engenharia de computadores I. Pinheiro, Wallace Anacleto. II Pinto, Raquel Coelho Gomes. III. Título. IV. Instituto Militar de Engenharia.

INSTITUTO MILITAR DE ENGENHARIA

Maj EDGARD HONORATO CARDOSO BERNARDO

ARQUITETURA PARA SUPORTAR SOBRECARGAS MOMENTÂNEAS EM AMBIENTE DE COMPUTAÇÃO EM NUVEM

Dissertação de Mestrado apresentada ao Curso de Mestrado em Sistemas e Computação do Instituto Militar de Engenharia, como requisito parcial para obtenção do título de Mestre em Ciências em Sistemas e Computação.

Orientador: Prof Wallace Anacleto Pinheiro - D.Sc.

Orientadora: Profª Raquel Coelho Gomes Pinto - D.Sc.

Aprovada em 12 de agosto de 2014 pela seguinte Banca Examinadora:

_____

Prof. Wallace Anacleto Pinheiro - D.Sc. do IME - Presidente

_____

Profª. Raquel Coelho Gomes Pinto - D.Sc. do IME

_____

Prof. Julio Cesar Duarte - D.Sc. do IME

_____

Prof. Lauro Luis Armondi Whately - D.Sc. da UFRJ

Rio de Janeiro

2014

Ao meu pai Pedro Bernardo que tanto me amou e incentivou e se faz presente em todos os momentos de minha vida, como exemplo de dedicação, vontade, espírito de luta, saudade e amor incondicional à família e ao trabalho.

AGRADECIMENTOS

Primeiramente agradeço a Deus, pela saúde e a paz interior, elementos essenciais para realização deste trabalho. À família representada por minha mãe Ely Cardoso Bernardo, eterna luz que ilumina meus caminhos, meu maior exemplo de vida, ao meu saudoso pai Pedro Bernardo (*in memoriam*), exemplo de luta e perseverança, cujos ensinamentos pautam minha conduta. Àos meus irmãos Maxwel e Marcello, amigos de todas as horas, que me impulsionam na caminhada e a querida Mariléia, parceira e amiga, sempre presente, com seu carinho e compreensão.

Aos professores e orientadores Wallace Anacleto Pinheiro e Raquel Coelho Gomes Pinto, pelos ensinamentos, pela amizade, pela confiança, pelo constante apoio na elaboração deste trabalho, sem os quais seria impossível a realização deste empreendimento científico. Espero que vocês continuem iluminando os alunos com seu conhecimento, força de espírito e dedicação. Meus sinceros agradecimentos.

Ao professor Julio Cesar Duarte, e ao professor Lauro Luis Armondi Whately, por terem aceitado fazer parte da banca examinadora deste trabalho.

Ao Sr Gen Div Emilio, meu comandante e chefe direto, cujo incentivo, confiança e exemplo, me motivaram ao empreendimento desta jornada academica.

Ao CEL Mauro Vasconcellos, agradeço pelo estímulo ao aprimoramento técnico e profissional, pelo apoio incondicional à pesquisa e pelo incentivo nos momentos de maior cansaço, seu apoio foi essencial para conclusão deste trabalho.

Ao TC Salles, Coordenador do Curso de Pós-Graduação em Sistemas e Computação, agradeço pelas primeiras orientações, pelo estímulo à pesquisa e pelo apoio nos momentos que se fizeram necessário.

Aos demais professores que fizeram parte da equipe docente do curso de Mestrado em Sistemas e Computação do Instituto Militar de Engenharia, pela valorosa contribuição em minha formação.

Aos integrantes da SE8, em especial ao funcionário civil Luiz Duarte, do Laboratório de Redes, por todo apoio dispensado. Sem a sua ajuda, as inúmeras configurações necessárias nos três laboratórios utilizados seriam muito mais complicadas.

Aos colegas Pedro Ferreira e Yan Martins, pelo apoio em vários momentos deste trabalho, suas contribuições foram fundamentais para construção da arquitetura proposta e para a realização dos experimentos realizados nesta pesquisa.

Ao amigo Luis Claudio Batista da Silva, companheiro de disciplina, de elaboração de artigos e seminários, seus conhecimentos, suas sugestões somadas e sua grande capacidade de trabalho foram alguns dos principais fatores que possibilitaram a melhoria deste projeto.

Aos colegas de turma que tornaram a sala de aula um ambiente familiar, de mútua colaboração, camaradagem e harmonia. Em especial aos amigos Marcus Albert, Patrick Lara, Tanilson Dias, Diego Esteves pelas grandes jornadas de estudos, pelo aprendizado em equipe e pelo apoio nos momentos de maior cansaço.

Aos colegas de laboratório Luciene e Phillipe pela ajuda e orientações que contribuíram muito para configuração do ambiente de desenvolvimento, pela amizade e apoio nas muitas horas de trabalho e estudo.

Aos companheiros da Divisão de Telemática do IME pelo apoio e incentivo nos momentos em que mais foram necessários.

A todos aqueles que escaparam a estas linhas, mas certamente tiveram uma importante parcela de contribuição neste projeto.

Edgard Honorato Cardoso Bernardo

“...Tudo vale a pena
Se a alma não é pequena.
Quem quer passar além do Bojador.
Tem que passar além da dor.
Deus ao mar o perigo e o abismo deu,
mas nele é que espelhou o céu.”
(Fernando Pessoa)

SUMÁRIO

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS

API	-	Application Programming Interface
ANS	-	Acordo de Nível de Serviço
CPU	-	*Central Processing Un*it (Unidade Central de Processamento)
IaaS	-	Infrastructure as a Service (Infraestrutura como serviço)
IP	-	Internet Protocol
GB	-	Gigabyte
MB	-	Megabytes
PaaS	-	Platform as a Service (Plataforma como Serviço)
TCP	-	Transmission Control Protocol
TI	-	Tecnologia da Informação
SaaS	-	Software as a Service (Software como Serviço)
UDP	-	User Datagram Protocol
MMV	-	Monitor de Máquinas Virtuais
VOLTAIC	-	Volume Optimization Layer To Assign Cloud resource
SVP	-	Servidor Virtual Privado

RESUMO

Este trabalho apresenta uma arquitetura computacional capaz de suportar os efeitos produzidos por sobrecargas momentâneas em servidores físicos e virtuais hospedados em ambiente de computação em nuvem. A arquitetura proposta objetiva a automatização da gerência de máquinas virtuais hospedadas neste ambiente, combinando uma estratégia proativa, que realiza o balanceamento de carga nos momentos em que não há sobrecarga das máquinas físicas e/ou virtuais, com uma estratégia reativa, que é acionada caso ocorra sobrecarga nestas máquinas. Em ambas as estratégias, é observado o acordo de nível de serviço (ANS) estabelecido para cada serviço hospedado de acordo com o modelo de infraestrutura como serviço (IaaS). As principais contribuições deste trabalho são a proposição em uma nova abordagem para tratamento de sobrecargas momentâneas neste ambiente e a implementação de uma arquitetura computacional, chamada Phoenix, capaz de realizar o tratamento de sobrecargas momentâneas, observando os ANSs dos serviços de infra-estruturas hospedados (IaaS), considerando os recursos CPU, memória e rede das máquinas físicas e virtuais. Os resultados comprovam que a arquitetura Phoenix é eficaz e possui destacada atuação no tratamento de sobrecargas de rede de máquinas virtuais, onde conseguiu realizar o isolamento da sobrecarga momentânea na máquina física hospedeira evitando a propagação de seus efeitos no cluster.

ABSTRACT

This work presents a computational architecture capable of withstanding the effects produced by the transient overloads on physical and virtual servers hosted on cloud computing environment. The proposed architecture aims at automating management of virtual machines that are hosted in this environment, combining a proactive strategy, which performs load balancing in times when there is no overload of physical and/or virtual machines with a reactive strategy, which is triggered in the event of overload on these machines. In both strategies, it is observed the service level agreement (SLA) established for each hosted service according to the model of infrastructure as a service (IaaS). The main contributions of this paper are the proposition of a new approach for treating momentary overloads in this environment and the implementation of a computing architecture, called Phoenix, capable of treating momentary overloads, watching the SLAs of infrastructure services hosted (IaaS), considering the CPU, memory and network of physical and virtual machines resources. The results show that the Phoenix architecture is effective and has outstanding performance in handling overloads virtual machine network, which has achieved the isolation of momentary overload on the physical machine host preventing the propagation of their effects on the cluster.

.

1 INTRODUÇÃO

A computação em nuvem pode ser definida como um tipo de sistema paralelo e distribuído, constituído a partir de uma coleção de computadores virtualizados e interconectados. Estes são dinamicamente disponibilizados e apresentados como recursos computacionais unificados, cujos serviços são baseados em acordos de nível de serviço (ANSs) (BUYYA et al., 2009).

Ao mover os serviços de TI para a nuvem, as organizações garantem maior flexibilidade e podem usar os serviços em um modelo sob demanda, os quais são pagos de acordo com a sua utilização. Neste modelo, toda infraestrutura de TI necessária ao suporte dos negócios é confiada ao provedor de serviços em nuvem, que garante seu funcionamento baseado em acordos de nível de serviço (ANSs), que são acordos formais entre provedores e consumidores do serviço, onde são estabelecidos os requisitos e responbilidades que devem ser observados por ambas as partes.

O crescimento da importância da computação em nuvem pode ser comprovado pelo interesse de grandes empresas que já oferecem soluções proprietárias como a Amazon, a Google, a Microsoft e a EMC. Do mesmo modo, a comunidade *open source* desenvolve soluções abertas que suportam nuvens como o OpenNebula e o Eucalyptus (CARVALHO, 2012).

Na nuvem, são realizados ajustes na infraestrutura que garantem os recursos computacionais necessários à permanência do serviço em funcionamento. Estes ajustes são realizados por meio da elasticidade, que pode ser definida como o grau de adaptação de um sistema computacional às variações na carga de trabalho por meio do provimento e do desprovimento de recursos (HERBST; KOUNEV; REUSSNER, 2013).

Deste modo, a computação em nuvem permite que recursos computacionais sejam entregues aos clientes em forma de serviços, os quais recebem a concessão de uso de recursos remotos, sem necessariamente adquiri-los. Neste modelo, podem ser contratados dinamicamente recursos computacionais (processamento, memória, armazenamento e rede) que podem ser implementados por meio uma tecnologia conhecida como virtualização (CARISSIMI, 2008).

Além do isolamento das aplicações, a virtualização permite a reconfiguração dos recursos virtualizados em uma máquina física, sendo que estes recursos podem ser aumentados, diminuídos ou mesmo migrados para outras máquinas físicas, conforme a dinâmica da carga de trabalho, de forma transparente para aplicação. Esta funcionalidade é oferecida por monitores de máquinas virtuais (MMV) ou hipervisores.

Porém, mesmo em um ambiente virtualizado, proporcionado pela computação em nuvem, podem surgir ameaças extremamente prejudiciais ao funcionamento destes serviços. Períodos de indisponibilidade, falhas na segurança dos dados armazenados, queda de desempenho, dentre outros, podem comprometer seriamente a credibilidade dos provedores de serviços.

Além dessas ameaças, um grande desafio deste ambiente é a manutenção da elasticidade da infraestrutura para suportar os serviços em conformidade com os ANSs estabelecidos. Variações na demanda por recursos devem ser atendidas sem prejuízo de qualquer cliente.

Para que os provedores consigam garantir os ANSs, minimizando eventuais prejuízos causados por sobrecargas ao funcionamento dos ambientes virtualizados, eles precisam gerenciar seus recursos computacionais de modo eficaz. Em pequena escala, isto pode ser feito manualmente com um pequeno grupo de administradores, porém quando se trata de datacenters hospedados em nuvem, compostos por milhares de equipamentos, muitos dos quais heterogêneos, uma gerência manual não possibilita a correção de problemas dentro do tempo necessário (WOOD et al.,2009). Deste modo, os administradores de nuvens precisam de um conjunto de ferramentas que os auxiliem no processo de tomada de decisão, diminua a carga sobre o operador, diminua o tempo de reação e minimize a possibilidade de violações de ANSs (CARVALHO, 2012). Assim, estabelece-se a necessidade do desenvolvimento de mecanismos que automatizem processos de gerência e aumentem a eficiência dos provedores de serviços em nuvem.

A gerência eficiente destes recursos pode trazer inúmeros benefícios, destacando-se a manutenção dos serviços oferecidos dentro dos níveis acordados, com a melhor utilização dos recursos. Porém, por si só a administração destas máquinas representa um problema complexo. Cada

máquina virtual (MV) pode apresentar um padrão diferente de utilização de recursos e contribuir para desestabilização do ambiente (CARVALHO, 2012). A partir desta perspectiva, torna-se ainda mais relevante a distribuição otimizada destas máquinas virtuais entre as máquinas físicas (MFs) disponíveis. Este é um grande desafio e é fundamental para garantir a qualidade e o nível de serviço acordado.

A partir da observação deste cenário, verifica-se a necessidade de estratégias que visem o atendimento dos ANSs contratados, considerando o suporte à sobrecargas eventuais e momentâneas. Nestas estratégias, pode ser considerado um comportamento proativo, que realize o balanceamento de carga (distribuição otimizada) quando não há sobrecarga das máquinas físicas, combinado com um comportamento reativo, que entre em ação quando a sobrecarga acontece minimizando seus os efeitos.

Este trabalho propõe que o uso da combinação de estratégias proativas e reativas devem ser consideradas de modo a suportar sobrecargas momentâneas nos ambientes virtualizados estabelecidos pela computação em nuvem.

Ao realizar o balanceamento de carga, permite-se o suporte a sobrecargas de maneira proativa, pois uma vez que o *cluster* de máquinas físicas esteja com cargas equilibradas, evita-se a sobrecarga prematura. Entretanto, se os limites estabelecidos para as máquinas físicas forem alcançados, estratégias pontuais e emergenciais, voltadas para liberação de recursos para as máquinas afetadas devem ser efetuadas, considerando também a situação global do cluster.

Neste contexto, foram consideradas para comparação como o método proposto estratégias definidas em trabalhos como o VOLTAIC (CARVALHO, 2012) e Sandpiper (WOOD et al., 2009). O primeiro tem foco no balanceamento de carga entre os nós do cluster, porém não considera o isolamento da sobrecarga da máquina virtual afetada na máquina física hospedeira. O segundo se preocupa em tratar situações de sobrecarga das máquinas físicas e virtuais de modo preditivo, porém não analisa o equilíbrio das cargas dos nós que compõem o cluster.

1.1 OBJETIVOS

1.1.1 OBJETIVO GERAL

O objetivo desta pesquisa é propor uma arquitetura computacional para automatização da gerência de máquinas virtuais hospedadas em ambiente de computação em nuvem que suporte sobrecargas momentâneas, combinando uma estratégia proativa, que realiza o balanceamento de carga nos momentos em que não houver sobrecarga das máquinas físicas e ou virtuais, com uma estratégia reativa, realizada quando ocorrer sobrecarga. Em ambas as estratégias são considerados como parâmetros o uso de CPU, memória e rede. Esta arquitetura deve observar os ANSs contratados, diminuindo a possibilidade de impacto deste tipo de sobrecarga nos demais serviços hospedados na nuvem.

1.1.2 OBJETIVOS ESPECÍFICOS

Para o cumprimento do objetivo geral proposto para este trabalho, foram estabelecidos os seguintes objetivos específicos:

- Proposição e especificação de um sistema automatizado capaz de suportar sobrecargas transientes em ambiente de computação nuvem;
- Implementação do sistema proposto utilizando, preferencialmente, tecnologias que utilizem código aberto e software livre que possibilitem a reprodução do modelo proposto.
- Configuração de ambiente computacional virtualizado e realização de experimentos para validação do sistema proposto.
- Análise e discussão dos resultados.

1.2 ORGANIZAÇÃO DO TRABALHO

O presente estudo encontra-se organizado em seis capítulos. No Capítulo 2, é apresentada uma revisão de literatura em torno de temas como computação em nuvem, elasticidade, virtualização, sobrecarga transiente, balanceamento de

carga, acordo de nível de serviço, e *oversubscription*. No Capítulo 3, são apresentados os trabalhos relacionados à pesquisa. O Capítulo 4 descreve a arquitetura proposta e sua implementação. O Capítulo 5 descreve os experimentos realizados e a análise de seus resultados. Por fim, o Capítulo 6 apresenta as considerações finais e indicações de trabalhos futuros.

.

2 CONCEITOS FUNDAMENTAIS

2.1 COMPUTAÇÃO EM NUVEM

A computação em nuvem é um modelo para acesso à rede sob demanda, ubíquo e conveniente para um pool compartilhado de recursos computacionais configuráveis que podem ser rapidamente provisionados e lançados com mínimo esforço de gerenciamento ou interação com o provedor de serviços (MELL; GRANCE, 2012).

O modelo de computação em nuvem visa prover acesso sob demanda a diferentes camadas da plataforma computacional, como por exemplo: redes, servidores, armazenamento, aplicações, serviços, etc. Estas funcionalidades podem ser rapidamente fornecidas ou liberadas pelo provedor de serviços. A nuvem tem por objetivo fornecer alta disponibilidade, elasticidade e possui cinco características essenciais (MELL; GRANCE, 2012):

1. Auto-atendimento: o consumidor configura cada recurso computacional conforme sua necessidade, sem exigir interação humana com os provedores de serviço;
2. Amplo acesso à rede: os recursos são disponibilizados na rede e acessados por meio de mecanismos padronizados. Isto possibilita o uso em diferentes dispositivos como celulares, *tablets* e notebooks;
3. Pool de recursos: os recursos computacionais do provedor são agrupados. Isto permite servir múltiplos consumidores em um modelo multi-inquilino (*multi-tenant*), ou seja, os recursos físicos e virtuais compartilhados são distribuídos e/ou redistribuídos dinamicamente de acordo com a demanda do consumidor;
4. Elasticidade: capacidades em termos de recursos computacionais podem ser fornecidos e retirados rapidamente e, em alguns casos, automaticamente. A quantidade de recursos disponibilizados passa para o consumidor a impressão de que a nuvem possui uma infraestrutura ilimitada; e
5. Medição do uso dos serviços: sistemas em nuvem controlam e otimizam o uso de recursos fornecendo métricas de acordo com o tipo de serviço. Tanto o

provedor quanto o consumidor podem monitorar e controlar a utilização dos recursos de forma transparente.

2.1.1 MODELO DE SERVIÇOS NA NUVEM

Conforme descrito em Mell e Grance (2012) e Dillon et al. (2010), a computação em nuvem disponibiliza e classifica os tipos de serviços conforme se segue:

Software como Serviço (*Software as a Service* - SaaS) – neste serviço, o cliente não administra ou controla a infraestrutura da rede, bem como servidores, desktops, sistemas operacionais. São disponibilizados softwares com propósitos específicos para usuários, através da Internet, por meio do uso de um navegador (MELL; GRANCE, 2012). A infraestrutura de nuvem muitas vezes emprega sistema *multi-tenancy*, ou seja, diferentes consumidores de aplicativos na nuvem são organizados em um único ambiente lógico no qual são compartilhas instancias de determinadas aplicações (DILLON et al.,2010). Isto permite economia de escala e otimização em termos de velocidade, segurança, disponibilidade, recuperação de desastres e manutenção. São exemplos desse tipo de serviço: Dropbox, GoogleDocs, Gmail e outros serviços que possibilitam o uso de aplicação/software via web.

Plataforma como Serviço (PaaS – *Plataform as a Service*) – neste tipo de serviço, é oferecido para o cliente uma plataforma para desenvolvimento, testes e implementação para aplicações web. Segundo Dillon et al. (2010), este modelo permite o suporte a todo o ciclo de vida do software. O cliente não administra ou controla a infraestrutura de rede, bem como servidores, sistemas operacionais; apenas tem controle sobre algumas ferramentas usadas para o auxilio na construção de aplicações web (MELL; GRANCE, 2012). Assim, diferentemente do SaaS, que suporta e provê uma aplicação pronta, o PaaS provê uma plataforma para desenvolvimento dessas aplicações.

Infraestrutura como Serviço (IaaS – *Infraestructure as a Service*) – este é o modelo de serviço que permite maior controle dos recursos virtuais por parte do cliente. De acordo com Mell e Grance (2012), neste serviço, apesar do cliente

não gerenciar ou controlar a infraestrutura interna da nuvem, é permitido o acesso direto à infraestrutura contratada como: servidores, sistemas operacionais e armazenamento de dados. O IaaS baseia-se em técnicas de virtualização para compartilhar, integrar e prover os recursos computacionais. Segundo Dillon et al. (2010), a estratégia básica da virtualização é definir máquinas virtuais independentes (MV) que são isoladas de outras MVs. Esta estratégia é diferente do modelo *multi-tenancy*, que visa transformar o software aplicativo em várias instâncias que podem ser executadas por vários consumidores de software. Como exemplo de IaaS, são citados o AmazonEC2 e o Eucalyptus.

2.1.2 MODELO DE IMPLANTAÇÃO

Em relação ao acesso e a disponibilidade de ambientes de computação em nuvem, existem diferentes tipos de modelos de implantação. Os modelos de implantação da computação em nuvem podem ser divididos em: público, privado, híbrido e comunidade (MELL; GRANCE, 2012) e são descritos como se segue:

- Privado – neste modelo de implantação, a infraestrutura de nuvem é utilizada exclusivamente por uma organização, a administração pode ser feita local ou remotamente pela própria empresa ou por terceiros, são empregadas políticas de acesso aos serviços (SOUSA; MOREIRA; MACHADO, 2009).
- Público - neste modelo de implantação, a infraestrutura de nuvens é disponibilizada para o público em geral e pode acessada por qualquer usuário que conheça a localização do serviço (SOUSA; MOREIRA; MACHADO, 2009). Neste modelo de implantação, não podem ser aplicadas restrições de acesso como técnicas de autenticação e autorização. (SOUSA; MOREIRA; MACHADO, 2009).
- Comunidade – neste modelo de implantação, diversas empresas compartilham uma nuvem que é suportada por uma comunidade específica que compartilham interesses como: missão, requisitos de segurança, política e considerações sobre flexibilidade (SOUSA;

MOREIRA; MACHADO, 2009). Este modelo pode implantado localmente ou remotamente e pode ser administrado por alguma empresa da comunidade ou por terceiros (SOUSA; MOREIRA; MACHADO, 2009).

- Híbrido - neste modelo de implantação duas ou mais nuvens, que podem ser: privadas, da comunidade ou pública se juntam e permanecem como entidades únicas e ligadas por uma tecnologia padronizada ou proprietária que permite a portabilidade de dados e aplicações (SOUSA; MOREIRA; MACHADO, 2009).

2.1.3 PAPÉIS DESEMPENHADOS NA COMPUTAÇÃO EM NUVEM

A computação em nuvem pode ser melhor entendida se houver possibilidade de classificar os atores dos modelos de acordo com os papéis desempenhados (BRISCOE; MARINOS, 2009). A FIG. 2.1 destaca estes papéis.

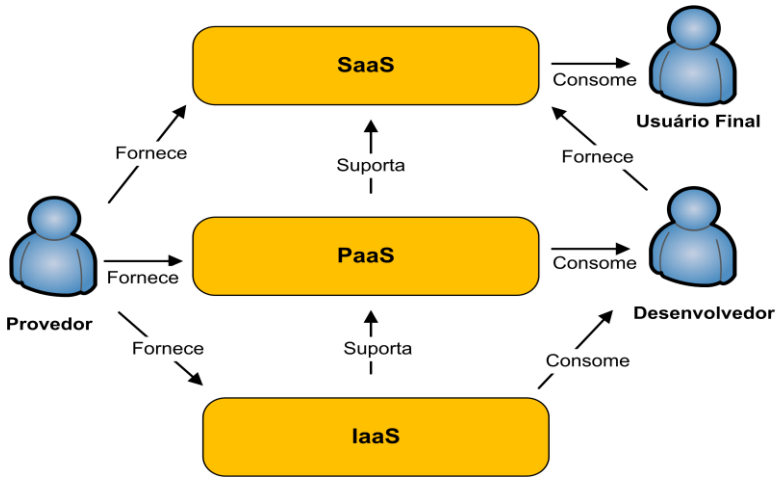

FIG. 2.1 Papéis na computação em nuvem (BRISCOE; MARINOS, 2009)

Em Sousa; Moreira; Machado (2009), é descrita a função dos principais atores envolvidos nos modelos de serviços oferecidos pela computação em nuvem. São eles: o provedor, o desenvolvedor e o usuário final. Estes podem ser descritos conforme se segue:

- O provedor é responsável por disponibilizar, gerenciar e monitorar toda a infraestrutura necessária a solução de computação em nuvem, retirando essas responsabilidades dos desenvolvedores e dos usuários finais. Fornece serviços nos três modelos apresentados (SaaS, PaaS, IaaS);
- Os desenvolvedores utilizam os recursos em termos de infraestrutura computacional, desenvolvem soluções e provêm serviços para os usuários finais;
- Os usuários finais são a última instância de consumidores dos serviços oferecidos pela computação em nuvem.

De acordo com seus os interesses, os atores podem realizar vários papéis ao mesmo tempo, porém, somente o provedor fornece suporte a todos os modelos de serviços (SOUSA; MOREIRA; MACHADO, 2009). Por exemplo, os desenvolvedores podem ser clientes de infraestrutura como serviço (IaaS) e fornecerem software como serviço para os usuários finais.

Ainda do ponto de vista de interação entre os três modelos de serviços, Sousa; Moreira; Machado (2009) esclarecem que o modelo IaaS fornece recursos computacionais, seja de hardware ou software, para a PaaS, que por sua vez fornece recursos, tecnologias e ferramentas para o desenvolvimento e execução dos serviços implementados, que podem ser disponibilizados no modelo de SaaS.

2.1.4 ARQUITETURA DE COMPUTAÇÃO EM NUVEM

A arquitetura de computação em nuvem é baseada em camadas, sendo que cada qual trata de uma particularidade na disponibilização de recursos para as aplicações (BUYYA et al., 2009) apud (SOUSA; MOREIRA; MACHADO, 2009).

Uma camada é uma divisão lógica de componentes de hardware e software, onde alguns destes recursos computacionais podem ser agrupados e organizados visando à realização de uma tarefa específica do sistema (SOUSA; MOREIRA; MACHADO, 2009).

Cada camada pode ter seu gerenciamento ou monitoramento de forma independente das demais, melhorando a flexibilidade, reusabilidade e escalabilidade no tocante a substituição ou adição de recursos computacionais sem interferir nas outras camadas (SOUSA; MOREIRA; MACHADO, 2009). A FIG. 2.2 exibe essas camadas e seus relacionamentos.

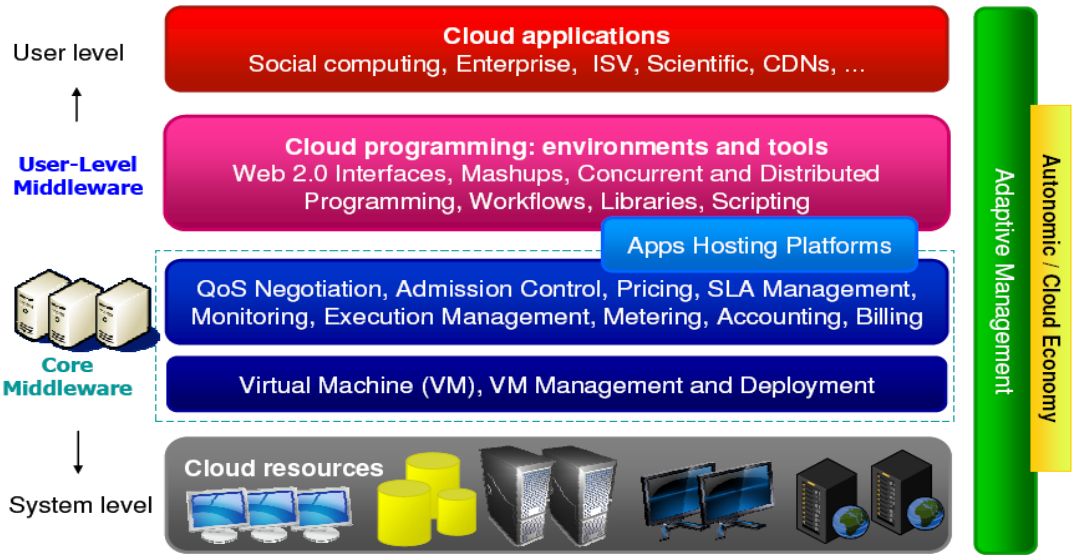

FIG. 2.2 Arquitetura da Computação em Nuvem (VECCHIOLA et al., 2009)

As camadas da computação em nuvem apresentadas na FIG. 2.2, podem ser resumidas, de baixo para cima, conforme se segue:

- Infraestrutura (*cloud resources*) - é a camada mais baixo nível na qual estão contidos datacenters, clusters, desktops e outros recursos de hardware, podendo ser estes recursos heterogêneos. Deste modo, há flexibilidade e facilidade de agregação de novos recursos à medida que se tornem necessários (SOUSA; MOREIRA; MACHADO, 2009).
- Middleware (*apps hosting platforms*) - é a camada responsável por gerenciar a infraestrutura física e tem por objetivo fornecer um núcleo lógico de uma nuvem. Estes serviços contêm negociações de QoS, gerenciamento do SLA, serviços de cobrança, serviços para verificar aceitação de requisições baseado no QoS e preço, serviços para

cálculo, serviços de gerenciamento de virtualização, entre outros (SOUSA; MOREIRA; MACHADO, 2009).

- Suporte a Construção de Aplicações (*cloud programing*) - esta camada contém ferramentas e/ou ambientes de desenvolvimento e é responsável por prover suporte ao desenvolvimento de aplicações. Os ambientes possuem interfaces Web 2.0, recursos de programação concorrente e distribuída, suporte a *workflows*, bibliotecas de programação e linguagens de programação (SOUSA; MOREIRA; MACHADO, 2009). Esta camada é utilizada pelos desenvolvedores de soluções no ambiente de computação em nuvem.
- Camada das Aplicações de Computação em Nuvem (*cloud applications*) - é a camada que fornece os aplicativos aos usuários.

2.2 ACORDO DE NÍVEL DE SERVIÇO

Os acordos de nível de serviço (ANS) podem ser definidos como contratos formais existentes entre os provedores de serviço e os clientes. Estes estabelecem regras que devem ser cumpridas por ambas as partes para validar a prestação e a utilização do serviço (CARVALHO, 2012).

O contrato de ANS tem por objetivo resguardar o fornecedor e cliente do serviço, buscando garantir que o serviço seja realizado de forma adequada e compatível com o que foi previamente acordado. Um ANS deve conter parâmetros objetivos e mensuráveis, os quais o fornecedor dos serviços se compromete a atender e restrições de uso as quais o cliente deve respeitar (FRANKE, 2010).

De acordo com Franke (2010), os componentes de um SLA se destacam na especificação dos serviços oferecidos e nos seus respectivos níveis de qualidade. Como um exemplo de ANS, pode-se definir que o provedor garanta a disponibilidade dos servidores em 99,99% do tempo, garanta que o cliente poderá utilizar um único processador, uma quantidade máxima de 1 GB de memória RAM e tenha a sua disposição até 100 Mbps de largura de banda de rede. Em contrapartida, o cliente não pode utilizar o serviço para prática de atividades ilícitas como invasão de sítios, por exemplo.

2.3 ESCALABILIDADE E ELASTICIDADE

É comum, a utilização do termo escalabilidade como sinônimo de elasticidade, no entanto, eles são conceitos diferentes e devem ser utilizados de forma própria.

A escalabilidade é **a capacidade de um sistema ser expandido** a um tamanho pré-estabelecido (estimado) de modo a se adaptar a crescentes e futuras cargas de trabalho ou demandas por recursos, mantendo o desempenho adequado, por meio da adição proporcional de novos de recursos, sem ater-se ao, no entanto, ao momento em que estes recursos serão efetivamente utilizados (AGRAWAL et al., 2011; GALANTE; BONA, 2012; SOBESLAVSKY, 2011)

A escalabilide de recursos, pode ser estabelecida de duas maneiras (dimensões) diferentes (WEBER *et al.*,2014):

- Escalabilidade vertical (*scale up/down*) refere-se à variação da quantidade de recursos por adição/remoção em um único nó existente.
- Escalabilidade horizontal ou (escale in/out) refere-se à variação da quantidade de recursos por adição/remoção de nós de um cluster.

A escalabilidade pode ter sua eficácia comprometida se o crescimento real da carga de trabalho não corresponder à estimativa. A expanção da capacidade implica no aumento dos recursos físicos, e pode ser difícil de reduzir a capacidade depois de uma expanção (SOBESLAVSKY, 2011).

Neste senário, cabe destacar que a escalabilidade não tem como foco o aspecto temporal da disponibilização de recursos. No contexto da computação em nuvem, a escalabilidade não faz qualquer suposição sobre quando os recursos são escalados. Descreve, no entanto, a quantidade de recursos adicionais que um sistema precisa para ser capaz de manter um nível de serviço constante (WEBER et al., 2014).

Uma das principais razões para a utilização da Computação em Nuvem é a possibilidade de fornecimento e aquisição de recursos de maneira dinâmica (COUTINHO; SOUSA, 2013).

A elasticidade é capacidade um sistema se adaptar as mudaças na carga de trabalho por meio do provisionamento de recursos de forma autônoma, de modo que os recursos disponíveis correspondam à demanda, no momento em que esta se apresenta (HERBST; KOUNEV; REUSSNER, 2013).

Do ponto de vista do consumidor, esta capacidade parece ser infinita, pois permite a aquisição de recursos computacionais, de acordo com a necessidade e tem sido usada principalmente para evitar a degradação do desempenho dos sistemas (GALANTE; BONA, 2012; SOUSA; MOREIRA; MACHADO, 2009). Além disso, recentes estudos têm descrito o uso da elasticidade para outros fins, tais como: o aumento da capacidade de recursos locais (CALHEIROS *et al.*, 2012; FITÓ *et al.,* 2010; MARSHALL *et al.,* 2010), redução de custos (SHARMA; SRIVASTAVA, 2013) e economia de energia (SHEN et al., 2011).

Com relação ao provedor, a elasticidade garante uma melhor utilização dos recursos computacionais, proporcionando economias de escala que viabilizam o atendimento de vários usuários ao mesmo tempo, mantendo a conformidade com os ANS estabelecidos (GALANTE; BONA, 2012;COUTINHO; SOUSA, 2013; MELL; GRANCE, 2012).

A elasticidade tem como pré-requisito a escalabilidade. Enquanto a escalabilidade estabelece a capacidade do sistema ser expandido, a elasticidade pressupõe a capacidade dos recursos se ajustarem à carga necessária, estando diretamente relacionada ao momento em que demandas reais por recursos serão efetivamente atendidas (HERBST et al., 2013; COUTINHO ; SOUSA, 2013).

Para implementar a escalabilidade na nuvem são utilizados os seguintes métodos (GALANTE; BONA, 2012): replicação, redimensionamento e migração:

- Replicação - consiste da adição/remoção de instâncias que são alocadas para determinado usuário (escalabilidade horizontal). As instancias (réplicas) podem ser MVs, containers ou módulos de aplicação no caso do modelo de serviço SaaS. Este método é utilizado pela maioria dos provedores e discutido por vários trabalhos, conforme apresentado por Galante e Bona (2012). Características

adicionais, como o balanceamento de carga que distribui a demanda pelas várias réplicas disponíveis, são suportadas por este método.

- Redimensionamento (escalabilidade vertical) – este método consiste na adição/remoção de recursos como processamento, memória e recursos de armazenamento a uma instância virtual (MV). Estes recursos adicionais podem ser explorados pelas aplicações hospedadas. O ElasticVM (DAWOUD; TAKOUNA; MEINEL, 2011), o PRESS (GONG; GU; WILKES, 2010) e Cloudscale (SHEN et al., 2011) são exemplos de sistemas que implementam este tipo de método.
- Migração - a migração de MVs descreve a transferência de uma máquina virtual em execução em uma máquina física para outra visando uma otimização da infraestrutura local ou global (WEBER et al., 2014). A elasticidade pode ser implementada pela migração de uma máquina virtual para uma máquina física que melhor se adapte à carga da aplicação (SHARMA et al., 2011) ou, por meio da consolidação e desconsolidação de um conjunto de máquinas em um host de servidor único (KNAUTH; FETZER, 2011). A quantidade recursos atribuídos a MV normalmente muda, enquanto o número de instâncias virtuais permanece inalterado, a migração de MVs então pode ser tratada como um caso especial de escalabilidade vertical (WEBER et al., 2014).

Comparando as duas definições, é possível identificar as diferenças fundamentais entre escalabilidade e elasticidade. A escalabilidade é uma propriedade estática que estabelece capacidade do sistema ser expandido quando necessário, até determinado limite, como por exemplo: servidores de que suportam até milhão de solicitações por minuto. Por outro lado, a elasticidade é uma propriedade dinâmica que permite que o sistema tenha a sua capacidade aumentada ou diminuída conforme a demanda, sendo que esta última está vinculada ao momento em que a demanda por recursos deverá ser atendida. Deste modo, a elasticidade resolve a lacuna deixada pela

escalabilidade no diz respeito ao momento em que deve realizada a adição e a remoção dos recursos.

2.4 VIRTUALIZAÇÃO

A virtualização pode ser definida com uma metodologia para divisão dos recursos de hardware de um computador em múltiplos ambientes de execução, onde são aplicados um ou mais conceitos ou tecnologias tais como particionamento por hardware ou software, *time-sharing*, simulação parcial ou total da máquina física, emulação, entre outros (PADHY et al, 2011).

Como resultado da aplicação dessas técnicas, são criadas máquinas virtuais (MVs) que fornecem ao usuário um ambiente completo muito similar a uma máquina física (MF). Estas MVs são abstrações do hardware do computador que permite que uma única MF possa agir como se fosse muitas máquinas.

Uma abstração é uma interface simplificada e homogênea para acesso aos recursos (hardware ou software) de um computador. Como exemplo, pode-se citar o sistema de arquivos que é disponibilizado para as aplicações. Este aparece como um conjunto de pastas e arquivos de tamanho variável onde as informações podem ser lidas e gravadas pelas aplicações. Os programadores de aplicativos podem então criar, gravar e ler arquivos sem saber detalhes da construção e da organização física do disco rígido.

As abstrações escondem detalhes de implementação de nível inferior, reduzindo assim a complexidade do processo de especificação do acesso aos recursos.

Porém, mesmo interfaces bem definidas têm suas limitações. Subsistemas e componentes projetados para um tipo de interface não funcionam com outra (SMITH; NAIR, 2005). Esta falta de interoperabilidade pode ser um limitador.

O uso de interfaces bem definidas também limita sua portabilidade, uma vez que os componentes desenvolvidos para uma interface não conseguem operar com uma interface distinta. Como exemplo, existem processadores de vários tipos com diferentes conjuntos de instruções, bem como Sistemas

Operacionais (SOs) com arquiteturas bem distintas. Aplicações que estão em código binário, teoricamente, só podem executar, de forma direta, a partir de um determinado conjunto de instruções e SOs específicos (FERNANDES, 2010).

A virtualização propõe uma solução para o problema da limitação da portabilidade. Ela pode ser descrita com uma abstração do sistema, que estabelece uma camada lógica na qual diferentes plataformas clientes podem ser executadas na mesma MF, aumentando significativamente a portabilidade. Esta camada lógica é denominada monitor de máquina virtual (MMV) ou hipervisor e fornece recursos virtualizados para a camada cliente que hospeda o SO convidado e suas aplicações.

A partir das considerações apresentadas por Padhy *at* al. (2011), pode-se descrever a composição de uma arquitetura básica de virtualização:

- SO convidado - é um sistema operacional que é executado em um ambiente virtual. Este utiliza os recursos de hardware alocados dinamicamente por meio do MMV ou software intermediário similar, usando interfaces padrão.
- Monitor de Máquina Virtual (MMV) - é responsável por gerenciar os recursos físicos compartilhando-os entre os SOs convidados. A camada de virtualização pode informar para cada SO convidado a existência de hardware diferente daquele realmente existente. A principal tarefa do hipervisor é lidar com a alocação de recursos e memória para as máquinas virtuais, garantindo que elas não interfiram umas nas outras, além disso, deve fornecer interfaces para administração de nível superior e ferramentas de monitoramento (ANWER *et al.*, 2010)
- SO Hospedeiro – é o SO da MF que irá suportar outros múltiplos sistemas operacionais. O SO hospedeiro acessa os recursos da MF como CPU, memória, dispositivos de rede, e aloca estes recursos para SOs convidados, conforme a necessidade.

A FIG. 2.3 representa a Arquitetura Básica da Virtualização, destacando seus componentes. A seta pontilhada indica a possibilidade do acesso direto ao

hardware feito por um determinado tipo MMV (Tipo I) que será apresentado nos próximos itens.

FIG. 2.3 Arquitetura Básica da Virtualização

2.4.1 TERMINOLOGIA UTILIZADA NA VIRTUALIZAÇÃO

De modo a facilitar o entendimento das técnicas de virtualização é necessário primeiramente identificar a terminologia utilizada:

- Máquina Hospedeira - é a máquina física que executa o software de virtualização. Ela contém os recursos físicos, como memória, espaço em disco e CPU o acesso à rede, que as máquinas virtuais utilizam.
- Máquina Virtual - é a representação virtualizada de uma MF que é executada e mantida pelo software de virtualização. Cada máquina virtual é implementada como um arquivo único ou uma pequena coleção de arquivos em uma pasta única no sistema hospedeiro. Seu comportamento é idêntico a uma máquina física não virtualizada. Cada máquina virtual possui um ambiente isolado e dedicado, sendo que todas compartilham os mesmos recursos

físicos por meio do hipervisor. Deste modo, pode-se controlar dinamicamente a entrega de recursos físicos para cada serviço hospedado.

- Software de virtualização - também conhecido como MMV ou hipervisor é um software que permite ao usuário executar máquinas virtuais em uma máquina hospedeira. Este é comumente usado em ambientes de nuvem para criar máquinas virtuais (servidores virtuais, discos de armazenamento virtual e, às vezes, para criar redes virtuais em centros de dados) e fornece às máquinas virtuais interfaces semelhantes às interfaces reais de hardware (CARVALHO, 2012).

A virtualização é implementada por meio de uma camada de software, podendo ser também auxiliada por hardware. Desta forma, um computador físico executa diversos computadores lógicos ou máquinas virtuais em paralelo (CARVALHO, 2012).

O MMV é executado em modo supervisor, enquanto que as máquinas virtuais em modo usuário. Quando estas tentam executar uma instrução em modo privilegiado, é gerada uma interrupção e o MMV se encarrega de emular a execução (MATTOS, 2008).

2.4.2 CLASSIFICAÇÃO DOS HIPERVISORES

A implementação de MMVs pode ser categorizada em de dois tipos: MMV tipo I e tipo II (PADHY et al, 2011). Estes possuem características próprias que os diferenciam e estabelecem sua aplicabilidade.

O MMV Tipo 1 também é conhecido como hipersor nativo (*bare-metal*). Ele é executado diretamente sobre o hardware da máquina hospedeira e os SOs convidados executam na camada logo acima deste. Exemplos deste tipo de hipervisor incluem: VMware ESX, Citrix XenServer e o Hyper-V da Microsoft

O MMV Tipo 2 é executado em um camada acima de um SO Hopedeiro, deste modo, o SO convidado é executado somente na terceira camada. Exemplos de tipo de MMV incluem o VMware Workstation (BUGNION et al.,

2012) e Parallels Desktop da SWSoft e o KVM (KIVITY et al., 2007). A TAB 2.1 apresenta estas características.

TAB 2.1 Tipos de Hipervisores - Fonte - Padhy *et al.* (2011).

MMV Tipo I	MMV Tipo II:
É executado diretamente no hardware físico;	O MMV é executado como um aplicativo em um sistema operacional normal;
Não tem um sistema operacional sendo executado em um nivel inferior;	O sistema operacional controla os recursos de hardware reais chamados como sistema operacional hospedeiro;
Totalmente responsável pela programação e concessão dos recursos de sistemas entre as máquinas virtuais;	O SO hospedeiro não tem conhecimento do MMV tipo II, este é tratado como qualquer outro processo no sistema; O sistema operacional que é executado dentro do MMV Tipo II é referenciado como o sistema operacional do cliente;
Mais seguro que o Tipo II.	Menos seguro que o MMV Tipo I, porque qualquer vulnerabilidade de segurança que leve comprometimento do sistema operacional hospedeiro também dará o controle total do sistema operacional convidado.
Exemplo: VMware ESX (Enterprise), Xen;	Exemplo: KVM, VMware GSX (estação de trabalho), UML (Modo User- Linux); Parallels Desktop, VMWAREFUSION, VIRTUAL BOX;

2.4.3 TÉCNICAS DE VIRTUALIZAÇÃO

De acordo Padhy *et al.* (2011), os diferentes métodos de virtualização podem ser descritos. Este são apresentados conforme se segue.

2.4.3.1 VIRTUALIZAÇÃO TOTAL

Nesta abordagem o sistema operacional hóspede não precisa ser modificado para executar sobre o MMV. A virtualização total fornece ao sistema operacional visitante uma réplica do hardware subjacente que consiste em prover no MMV suporte a um conjunto genérico de dispositivos (MATTOS, 2008; CARISSIMI, 2008). Dessa forma, o sistema operacional visitante pode executar operações como se estivesse executando no hardware da máquina física (CARISSIMI, 2008). A FIG 2.4 apresenta a interação entre as camadas em uma arquitetura de virtualização total. A vantagem desta abordagem é que os sistemas operacionais podem ser executados diretamente em um ambiente virtual, sem modificações, Porém, o MMV precisa interceptar todas as instruções e “converter” o espaço de endereçamento do sistema hóspede para o endereço real (hospedeiro) (CARISSIMI, 2008).

O KVM (KIVITY et al., 2007) é um exemplo de MMV que utiliza esta técnica de virtualização.

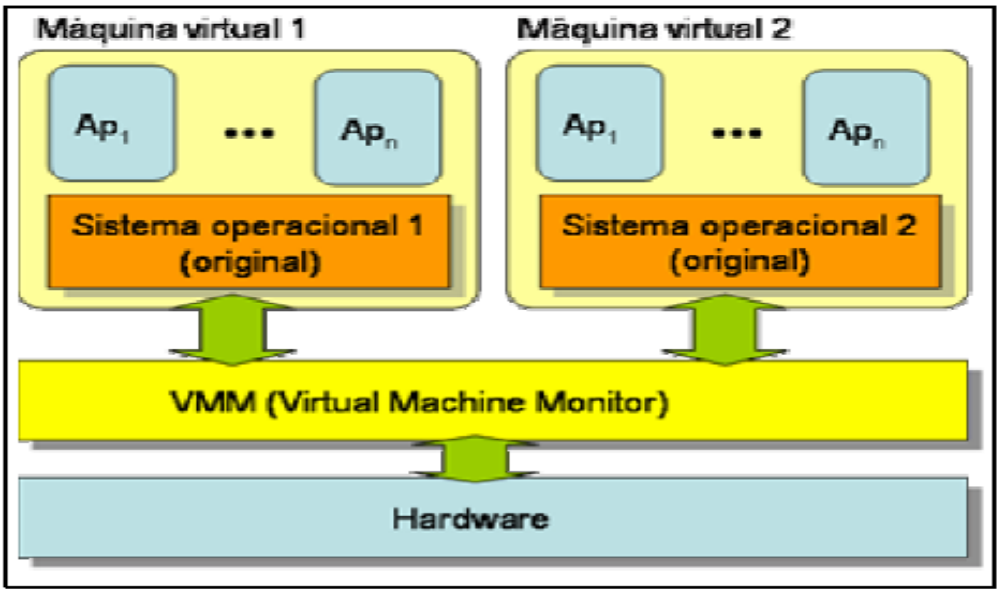

FIG. 2.4 Virtualização Total (CARISSIMI, 2008)

Nos últimos anos foram desenvolvidos processadores com extensão para virtualização no próprio hardware (ADAMS; AGESEN, 2006). Este fato contribuiu para melhoria de ferramentas que utilizam a implementação da virtualização total. São exemplos desta tecnologia o IVT (Intel Virtualization Technology), da Intel (UHLIG et al., 2005) e do AMD-V (AMD Virtualization), da AMD (AMD, 2005).

O KVM (KIVITY et al., 2007), tem como pré-requisito a existencia de uma destas tecnologias instaladas no processador da MF hospedeira.

2.4.3.2 PARA-VIRTUALIZAÇÃO

A para-virtualização tem como requisito a necessidade de alteração do sistema operacional do hospedeiro, para este possa executar instruções que precisem alterar o estado do sistema encaminhando-as diretamente ao MMV. Essa mudança representa um ganho de desempenho, uma vez que não será necessário testar todas as instruções solicitadas. A FIG. 2.5, possui uma parte hachurada que representa o kernel do sistema operacional compartilhado entre a máquina hospedeira e a visitante. Nesta abordagem, os dispositivos de hardware são acessados por drivers do próprio MMV (CARISSIMI, 2008).

O Xen (BARHAM et al., 2003) é um exemplo de MMV que utiliza esta técnica de virtualização.

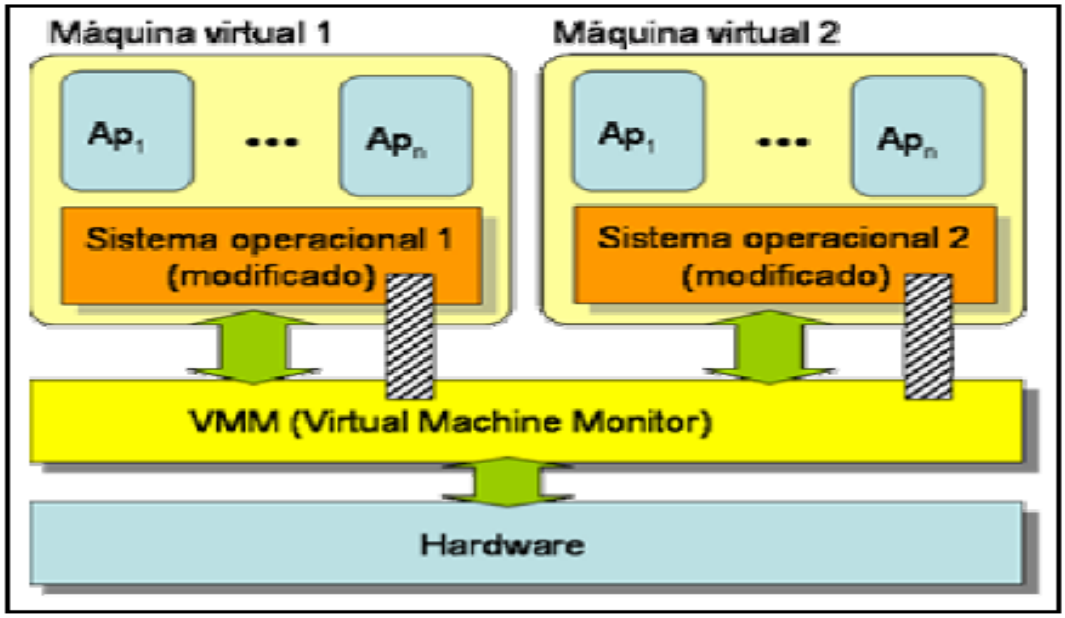

FIG. 2.5 Para-virtualização (CARISSIMI, 2008)

2.4.3.3 EMULAÇÃO

É o método de virtualização no qual uma arquitetura de hardware completa é criada por meio de software. Este software é capaz de replicar a funcionalidade de um processador e demais componentes da MF. Este método oferece uma grande flexibilidade uma vez que o SO convidado não precisa ser modificado para funcionar (PADHY et al., 2011).

A emulação apresenta desvantagens em termos de desempenho uma vez que cada instrução do SO convidado deve ser traduzida para ser entendido pelo SO Hospedeiro. O SO convidado permanece inalterado, mas é executado em uma CPU emulada por software. Este processo de conversão é extremamente lento em comparação com a velocidade do SO convidado rodando em um ambiente real.

Portanto, a emulação torna-se mais adequada nos casos em que a velocidade não é crítica, ou quando nenhuma outra técnica de virtualização puder ser usada.

2.4.3.4 VIRTUALIZAÇÃO REALIZADA POR MEIO DO SISTEMA OPERACIONAL (*CONTAINERS*)

Neste método de virtualização, o kernel do SO hopedeiro fornece várias instâncias que disponibilizam espaço para que o usuário possa rodar suas aplicações de forma isolada. Este modelo apresenta várias características dos métodos de virtualização tradicionais, no entanto, somente permite que aplicações de usuário (que sejam capazes de funcionar normalmente no sistema operacional hospedeiro) possam ser executadas (PADHY;et al., 2011).

Ao contrário de para-virtualização e da virtualização completa, a virtualização em nível de sistema operacional não conta com um hipervisor. Em vez disso, o sistema operacional é configurado para criar e isolar as múltiplas instâncias de um sistema operacional dentro de uma única máquina hospedeira (TUNCAY, 2010). As instâncias de SO convidado são muitas vezes referidas como servidores virtuais privados (SVPs).

A vantagem de utilizar a virtualização em nível de sistema operacional está principalmente no desempenho, uma vez que não é necessária a intervenção de

um hipervisor. Isso normalmente resulta no bom desempenho dos sistemas que utilizam esta funcionalidade. A principal desvantagem é que todas as instâncias SVPs compartilham um único kernel. Assim, se houver erros no kernel ou se o mesmo for comprometido, todas as instâncias de VPS são comprometidas. O Solaris Containers da Oracle descrito por Victor (2006) é um exemplo de virtualização de nível de sistema operacional.

2.4.3.5 VIRTUALIZAÇÃO NATIVA

A Virtualização nativa aproveita o suporte do hardware disponível no processador para viabilizar a virtualização. Ela permite que vários SOs convidados não modificados rodem em paralelo, desde que todos os sistemas operacionais sejam capazes de funcionar com o processador instalado na MF hopedeira.

A virtualização nativa é um meio termo entre a virtulização total e paravirtualização e não requer nenhuma modificação do SO convidado para melhorar o desempenho da virtualização. O MMV faz a tradução dos comandos do SO convidado para o hardware da MF hospedeira a qual disponibiliza os recursos necessários para cada SO convidado (PADHY; PATRA; SATAPATHY, 2011).

Esta configuração permite um aumento na velocidade do SO convidado por meio da aceleração em hardware, mas pode ocorrer uma degradação de desempenho nos momentos em que as instruções estivem requisitando ações que precisam ser emuladas pelo hipervisor ao invés de serem acessadas diretamente pelo hardware (GMACH et al., 2009).

Diferente da virtualização total, na virtualização nativa não ocorre a emulação do processador portanto, o uso de processadores que possuem aceleração para virtualização em hardware como a série x86 de 64 bits da Intel (Intel-VT) e da AMD (AMD-V) são essenciais.

2.5 SOBRECARGA TRANSIENTE

A sobrecarga momentânea, ou sobrecarga transiente, é o aumento súbito e de curta duração da carga de um sistema, para valores acima de sua capacidade. Eventos de sobrecarga transiente em servidores Web formados a partir de requisições legítimas dos usuários são de especial interesse deste trabalho. Eles são conhecidos como *flash crowds*, *hotspots* ou efeito *slashdot*, e se tornam cada vez mais comuns à medida que a utilização e o número de usuários de serviços oferecidos na Web aumentam (PAN; ATAJANOV, 2004).

O aumento dramático de tráfego legítimo na carga de um servidor Web, causa um grande congestionamento e perda de pacotes. Eventos de sobrecarga transiente são caracterizados por períodos curtos de carga intensa que se alternam com períodos longos de carga leve. A carga do servidor corresponde à taxa de chegada de requisições e estes eventos podem ser recorrentes em servidores Web (FARAH; MURTA, 2007).

As características das sobrecargas transientes foram analisadas por Pan e Atajanov (2004). Os autores sugerem que eventos de grande interesse ocasionam esse tipo de carga, e que o volume de requisições de objetos populares pode aumentar dramaticamente, chegando a atingir taxas 10 a 100 vezes maiores do que a taxa média observada no servidor. Esses eventos são de curta duração, o que indica que o superdimensionamento não é uma solução razoável, tendo em vista que o servidor será subutilizado na maior parte do tempo (FARAH; MURTA, 2007).

O crescimento do volume de carga é instantâneo, o que torna difícil a detecção da sobrecarga antes que ela ocorra. Além disso, Farah e Murta (2007) demonstram que recursos de CPU e largura de banda de rede são os primeiros a serem sobrecarregados. Os testes realizados mostram que a sobrecarga transiente, mesmo em pequena intensidade, prejudica significativamente o desempenho do servidor, diminuindo sua taxa máxima de serviço e registrando tempos de resposta inaceitáveis.

Eventos de sobrecarga em servidores web ocorrem frequentemente e tendem a aumentar à medida que o número de internautas e a demanda por serviços Web crescem.

2.6 BALANCEAMENTO DE CARGA EM AMBIENTE DE COMPUTAÇAO EM NUVEM

A computação em nuvem pode ser entendida como um grande sistema de computação paralela e distribuída que emprega recursos computacionais para entregar serviços sob demanda para os usuários finais. Neste sistema, é permitido que uma ampla gama de usuários tenham acesso a uma infraestrutura de hardware e software virtualizado, distribuído e escalável, via Internet (RAY; SARKAR, 2012).

O principal desafio da computação em nuvem é o controle de recursos ou o balanceamento de carga, uma vez que mesmo neste ambiente, caracterizado pelo compartilhamento de recursos, requisitos como tempo de resposta, confidencialidade e segurança devem ser garantidos (RAY; SARKAR, 2012).

Neste contexto, o balanceamento de carga pode ser definido como uma metodologia de distribuição de carga de trabalho entre vários computadores ou outros recursos ligados por rede com o objetivo de otimizar a utilização de desses recursos, maximizar taxa de transferência, mínimizar o tempo de resposta e evitar sobrecarga (KHIYAITA et al., 2012; RAY; SARKAR, 2012).

De forma mais abrangente, o balanceamento de carga pode ser visto com uma solução eficiente para várias questões relacionadas ao uso e a configuração do ambiente de computação em nuvem. Segundo Mayanka; Katyal (2013) o balanceamento de carga deve levar em conta duas grandes tarefas: a primeira é a eficiência na alocação de recursos, esta atividade define quais recursos estarão disponíveis para atender a requisitos do usuário. A segunda é o agendamento de tarefas que define a maneira como o recurso alocado estará disponível para o usuário final (ou seja, se o recurso estará totalmente disponível até a conclusão da tarefa ou se o recurso estará disponível em modo compartilhado no tempo ou no espaço). A correta execução destas tarefas assegura que (MAYANKA; KATYAL, 2013):

- Os recursos sejam facilmente disponibilizados de acordo com a demanda;

- Os recursos sejam utilizados com eficiência sob condições em que a carga do sistema esteja alta ou baixa;
- Haja economia de energia caso os usos dos recursos da nuvem estejam abaixo de certo limiar; e
- Haja redução do custo de utilização dos recursos.

2.6.1 DESAFIOS QUE ENVOLVEM ALGORITMOS DE BALANCEAMENTO DE CARGA

Existem alguns aspectos que devem ser observados na concepção de algoritmos que se propõem a tratar da questão do balanceamento de carga no ambiente de computação em nuvem. Em Nuaimi et al. (2012) e Swarnkar et al.(2013) são evidenciados alguns dos principais aspectos que devem ser observados na construção desses algoritmos. Estes são apresentados de forma resumida como se segue:

- Distribuição Espacial dos Algoritmos na Nuvem - alguns algoritmos são projetados para ser eficientes apenas na intranet ou em redes onde os atrasos de comunicação são insignificantes. No entanto, existe o desafio de se projetar algoritmos de balanceamento de carga que possam trabalhar com nós espacialmente distribuídos. A distância entre o cliente e os nós que estão processando a tarefa e as distâncias entre os nós envolvidos na prestação do serviço de processamento devem ser considerados (NUAIMI et al., 2012).
- Replicação/Armazenamento - A replicação completa não leva o um armazenamento eficiente, uma vez que os mesmos dados são armazenados em todos os nós replicados, aumentando os custos. O desafio então, é a replicação parcial dos conjuntos de dados em cada nó, com um certo grau de sobreposição. Isto pode levar a uma melhor distruibuição dos dados, ainda que haja o aumento da complexidade do algoritmo, que terá que considerar a distribuição dos dados em todos os nós da nuvem (NUAIMI et al., 2012).

- Complexidade de algoritmo - são preferidos algoritmos menos complexos em termos de implementação e operação. A maior complexidade de implementação pode causar alguns problemas de desempenho. Além disso, quando os algoritmos exigem mais informação e maior comunicação para monitoramento e controle, podem causar atrasos e problemas de eficiência (NUAIMI et al., 2012).
- Tolerância à falhas - considerando os diferentes nós que compõem a nuvem, o algoritmo de coleta de dados e balanceamento de carga deve ser concebido de forma a evitar tornar-se um ponto único de falha, sendo este o grande desafio (NUAIMI et al., 2012). Este ainda deve ter a capacidade de realizar a balanceamento carga uniforme apesar de uma falha arbitrária em nó ou no link (SWARNKAR; et al. , 2013)

2.6.2 MÉTRICAS PARA A AVALIAÇÃO

Existem algumas métricas que podem ajudar a avaliação e melhoria do comportamento de algoritmos que tratam da questão do balanceamento de carga no ambiente de computação em nuvem. Em Sidhu e Kinger (2013) são evidenciados algumas das principais métricas, que podem contribuir para avaliação desses algoritmos. Estes são apresentados de forma resumida como se segue:

- Sobrecarga associada - determina a sobrecarga gerada durante a execução de um algoritmo de balanceamento de carga. É composta pela execução de tarefas e comunicação entre processos. A sobrecarga deve ser minimizada para que o algoritmo de balanceamento carga possa trabalhar de forma eficiente (SIDHU; KINGER, 2013).
- Desempenho – é usado para verificar a eficiência do sistema. Deve ser melhorado de modo que seja obtido um custo razoável, por

exemplo, reduzir o tempo de resposta, mantendo atrasos aceitáveis (SIDHU; KINGER, 2013).

- Utilização de Recursos - é usado para verificar eficácia da utilização dos recursos. Ele deve ser otimizado para um eficiente balanceamento (SIDHU; KINGER, 2013).
- Escalabilidade - é a capacidade de um algoritmo para realizar equilíbrio de carga de um sistema com um número infinito de nós. Essa métrica deve ser melhorada a fim de atender a capacidade de crescimento da nuvem (SIDHU; KINGER, 2013).
- Tempo de resposta - é o tempo necessário para um algoritmo de balanceamento de carga responder a uma determinada demanda em um sistema distribuído. Este métrica deve ser minimizada (SIDHU; KINGER, 2013).

2.6.3 CLASSIFICAÇÃO DOS ALGORITMOS DE BALANCEAMENTO DE CARGA

Algoritmos de balanceamento de carga podem ser categorizados em diferentes classes baseadas na observação do estado atual do sistema, forma de execução do algoritmo e no agente que inicia o algoritmo.

De modo geral, os autores classificam os algoritmos de balancemanto como estáticos e dinâmicos, porém esta classificação se mostrou insuficiente para evidenciar a abordagem proposta neste trabalho.

De modo a tornar mais visível a classificação do algoritmo proposto no Capítulo 4, é proposta nesta Seção uma nova organização classificatória para os algoritmos de balanceamento de carga a partir das classificações propostas por Alakeel (2010), Khiyaita et al.(2012), Sidhu e Kinger (2013) e Rewehel e Mostafa (2014). A visão esquemática dessa classificação pode ser vista na FIG 2.6 e é descrita como se segue:

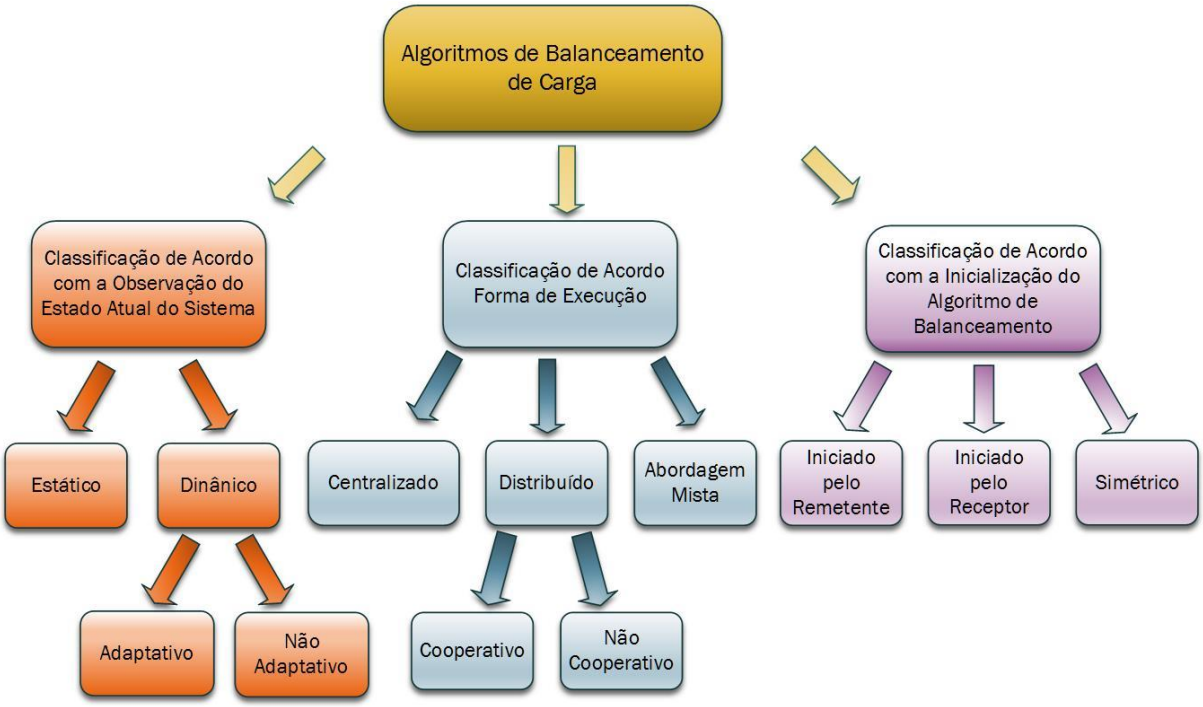

FIG. 2.6 Classificação dos Algoritmos de Balancemanto de Carga.

2.6.4 CLASSIFICAÇÃO DE ACORDO COM A OBSERVAÇÃO DO ESTADO ATUAL DO SISTEMA

Esta classificação foi estabelecida a partir da proposta de Alakeel (2010) e tem foco na capacidade do lagoritmo observar mudanças no estado atual no sistema. A partir desta perspectiva os algoritmos são classificados de duas formas: estático ou dinâmico e descritos como se segue:

- Abordagem estática: esta abordagem leva em consideração o conhecimento prévio da situação global do sistema, a existência de recursos para execução das rotinas e comunicação ente os nós (KHIYAITA et al., 2012). Algoritmos estáticos dividem a carga entre os nós da rede sem levar em consideração o estado atual do sistema.
- Abordagem dinâmica: esta abordagem leva em consideração o estado atual do sistema durante as decisões de balanceamento de carga. Esta abordagem permite o deslocamento da carga de um nó sobrecagarregado para outro que tenha condições de recebê-la.

Portanto, possui melhores condições para tomada de decisão de balanceamento. A habilidade para reagir conforme as mudanças de carga do sistemas é uma das grandes vantagens dessa abordagem (ALAKEEL, 2010). O balanceamento dinâmico pode ser executado de duas maneiras: adaptativa e não adaptativa. A primeira descreve a capacidade do algoritmo de adaptar a distribuição de carga de acordo com as alterações de status do sistema, alterando incluisive seus parâmetros e seus algoritmos dinamicamente. A segunda descreve a capacidade do algoritmo de adaptar a distribuição de carga de acordo com as alterações de status do sistema, porém as alterações em seus parâmetros e em seus algoritmos necessitam ser feitas pelo administrador. Esta subdivisão foi daptada a partir da classificação do balanceamento de carga quanto à topologia descrita por Khiyaita et al.(2012).

2.6.5 CLASSIFICAÇÃO DE ACORDO FORMA DE EXECUÇÃO

Esta classificação foi estabelecida a partir da proposta de Khiyaita et al.(2012), onde o foco está no modo como o algoritmo gerencia a demanda por balaceamento de carga. Esta é descrita como se segue:

- Abordagem Centralizada: nesta abordagem, um único nó é responsável por gerenciar a distribuição de carga dentro de todo o sistema a partir das informações dos nós do cluster;
- Abordagem Distribuída: nesta abordagem, cada nó, de modo independente constrói seu próprio vetor de carga a partir da coleta de informações dos outros nós. As decisões de distribuição de carga são tomadas localmente usando estes vetores (KHIYAITA et al., 2012). Adicionalmente, foi incluída nesta abordagem a subdivisão proposta por Alakeel (2010) que estabelece que o balanceamento dinâmico pode ser executado de duas maneiras cooperativa e não cooperativa. Na primeira, os nós trabalham juntos, dividindo tarefas para alcançar um objetivo comum, por exemplo, melhorar o tempo

de resposta de uma aplicação demanda por clientes externos). Na segunda, cada nó funciona de modo independentemente visando o atendimento de uma demanda local, por exemplo, melhoria do tempo de resposta de uma tarefa localizada em determinado servidor.

- Abordagem Mista: os nós do sistema são divididos em clusters, onde o balanceamento de carga é realizado de modo centralizado. Um nó central é eleito em cada cluster e gerencia o balancemento de carga entre os nós do cluster. Além disso, cada nó central dos clusters que compõe a nuvem mantém informações dos demais nós centrais. Neste nível, o balanceamento de carga é feito de forma distribuída. Portanto, o balanceamento de carga de todo o sistema é feito via os nós centrais da nuvem (ALAKEEL, 2010). Esta combinação permite tirar proveito das melhores características das duas abordagens anteriores.

2.6.6 CLASSIFICAÇÃO DE ACORDO COM A INICALIZAÇÃO

Esta classificação, conforme o proposto por Alakeel (2010) e Mishra e Jaiswal (2012), estabelece que o algoritmo de balancemento de carga também pode ser classificado de acordo com sua inicialização:

- Iniciado pelo Remetente: algoritmos que tomam decisões de balanceamento a partir do início ou da chegada de uma nova tarefa no nó que a está executando determinda tarefa são chamados de algoritmos iniciados pelo remetente, ou seja, a partir daquele que está sobrecarregado e necessita encontrar um outro nó que possa receber algumas de suas tarefas.
- Iniciado pelo Receptor: algoritmos que tomam decisões de balanceamento para apartir da finalização de uma tarefa são referenciados como iniciados pelo receptor, ou seja, iniciados por aqueles nós que estão subutilizados e tem condições de receber outras tarefas.

- Simétrica: É a combinação dos dois tipos anteriores. Hora é iniciada pelo remetente e hora é iniciada pelo receptor, podendo ser iniciada de acordo com o limiar estabelecido para cada inicialização. Este tipo de inicialiazação requer maior controle.

2.6.7 POLITICAS UTILIZADAS PELOS ALGORITMOS DE BALANCEAMENTO DE CARGA

Para realizar o balanceamento de carga em ambiente de computação em nuvem os algoritmos se utilizam de algumas políticas que de forma resuminda podem ser descritas a partir da aptação do trabalho realizado por Kaur e Lutra (2014):

- Politica da Informação: definie que informação é requerida, como é coletada e quando.
- Política do Disparo: definie quando o balanceamento de carga se inicia.
- Políta dos Recursos: definie que tipo de recursos estarão disponiveis durante o balanceamento de carga.
- Política da Locação: utiliza todos recursos disponibilizados pela Política dos Recursos para determinar quais MFs enviarão ou receberão MVs.
- Política da Seleção: é utilizada para encontrar a MV que irá ser transferida de uma MF sobrecarregada para uma com capacidade para recebê-la.

No ambiente de computação em nuvem os recursos computacionais devem estar disponíveis aos clientes conforme o ANS estabelecido. Um algoritmo de balanceamento de carga ideal deve evitar sobrecarga ou subutilização de qualquer nó específico hospedado neste ambiente (MOHARANA et al., 2013).

Há portanto, a necessidade de uma administração eficiente de recursos (RAY; SARKAR, 2012).

2.7 OVERSUBSCRIPTION

Os provedores de serviço de IaaS buscam maximizar seus lucros concentrando MVs em servidores físicos que não suportam toda a capacidade possível de ser utilizada pelas MVs. Esta prática é definida como o*versuscription* (BASET; WANG; TANG, 2012). Porém, ela pode levar a uma sobrecarga da máquina física e a um possível comprometimento do ANS. Um provedor de serviços em nuvem deve gerenciar a carga de suas máquinas físicas de modo a não comprometer os serviços hospedados (BASET; WANG; TANG, 2012).

Em uma infraestrutura como serviço (IaaS), o provedor permite que os clientes executem múltiplas máquinas virtuais hospedadas em suas máquinas físicas. O provedor de IaaS trabalha com múltiplos inquilinos, ou seja, hospeda varias MVs que executam cargas de trabalho de diferentes clientes ao mesmo tempo que também compartilham o hardware da uma mesma MF.

Para um provedor de nuvem maximizar os lucros, a chave é efetivamente mitigar sobrecargas e atender ANSs contratados junto aos clientes.

A máquina física gerencia recursos como CPU, memória, disco e recursos de rede. MFs acometidas por oversubscription podem sofrer sobrecarga em qualquer desses recursos.

A migração em tempo real (Live Migration) permite que um provedor migre um conjunto de MVs de uma MF sobrecarregada para uma MF menos carregada. A migração em tempo real é uma boa solução quando o armazenamento de MVs estam localizadas em um dispositivo hospedado em uma SAN (Storare Area Network) (BASET; WANG; TANG, 2012).

Existem duas etapas para manipulação do *oversubscription*: detecção e mitigação (BASET; WANG; TANG, 2012). Nos Capítulos 4 e 5, estas etapas serão discutidas com maior profundidade.

2.8 CICLO OODA

O ciclo OODA, definido por Boyd (1981), pode ser entendido como o conjunto de ações integrantes do processo decisório, composto pelas fases:

Observar, Orientar, Decidir e Agir (OODA), essenciais à eficiente aplicação dos processos relativos ao comando e controle.

Consiste em um modelo elaborado no intuito de possibilitar a compreensão do funcionamento da atividade de Comando e Controle (C^2), servindo como ferramenta de auxílio para a concepção e a avaliação de seus processos com destaque ao processo de tomada de decisão. Esta abordagem pode ser vista de forma esquemática na FIG. 2.7.

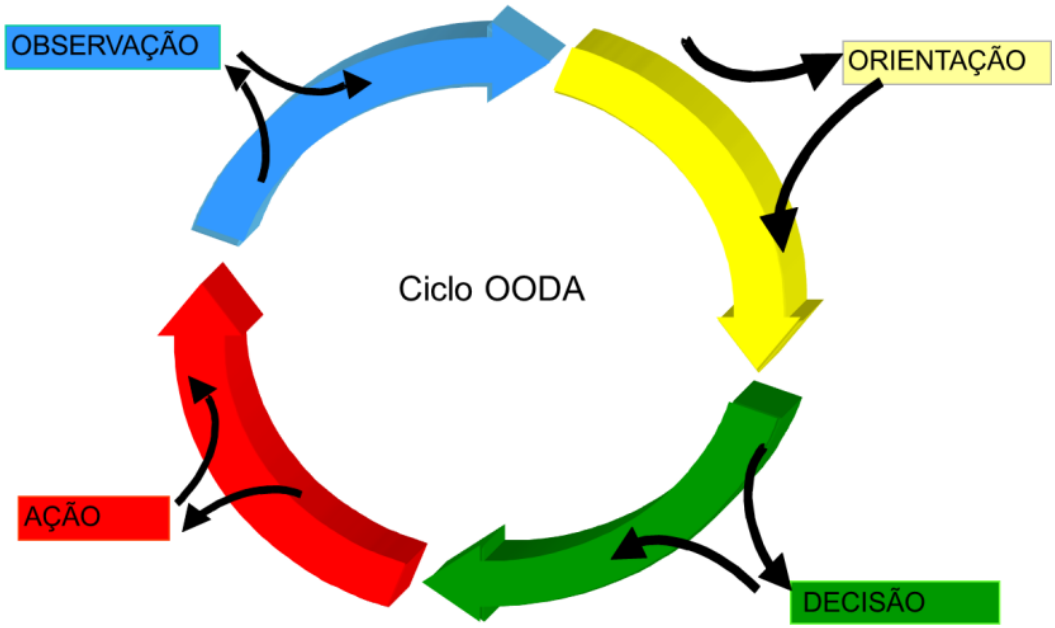

FIG. 2.7 Ciclo OODA adaptado de Boyd (1981)

De modo geral, o funcionamento do Ciclo OODA é descrito em fases como se segue:

- Fase de Observação – esta fase obtém coleta de dados de determinado campo de interesse.
- Fase de Orientação – esta fase busca adquirir consciência situacional, a partir dos dados coletados na fase de observação. São executadas as análises de diretrizes operacionais e são produzidas informações concretas e relevantes as quais são encaminhadas para tomada de decisão.

- Fase de Decisão – nesta fase, com base nas informações recebidas da fase de orientação, são identificados os atores, estabelecida a tática e emitidas às ordens que serão executadas.
- Fase de Ação – nesta fase, são empregados os meios, preparadas as ações contínuas e controladas as ações correntes com base na orientação da fase decisão.

Na primeira etapa, é percebida uma mudança no curso dos acontecimentos; na segunda, é construída uma imagem crítica da situação; na terceira, chega-se à decisão da conduta a ser adotada; e, na última, são implementadas as ações decorrentes da decisão tomada, voltando-se à da observação para um novo ciclo.

A dinâmica promovida pelo conjunto de ações, os processos estabelecidos em cada uma das quatro fases e os mecanismos de comando e controle propostos para o seu funcionamento inspiraram a arquitetura proposta, pois as fases do ciclo OODA possuem grande similaridade com os processos modelados para o gerenciamento de sobrecargas em ambiente de virtualizado, como será visto no Capítulo 4.

.

3 TRABALHOS RELACIONADOS AO TEMA PROPOSTO

Diversos trabalhos abordam o problema de alocação de máquinas virtuais (MVs) em servidores físicos. Porém muitas destas propostas focam no desafio de controlar a movimentação das MVs, sem considerar a questão da administração da escalabilidade baseada na real necessidade das máquinas virtuais e as considerações impostas pelo ANS contratado. Neste contexto, a administração de escalabilidade tem fundamental importância, pois permite que haja maior economia de recursos para o provedor de serviços uma vez que os recursos podem ser liberados gradativamente à medida que são necessários, de acordo com os níveis de serviços contratados. Neste cenário, dentre os principais trabalhos pesquisados, destacam-se: Sandpiper (WOOD et al., 2009) e VOLTAIC (CARVALHO, 2012).

3.1 SANDPIPER

O Sandpiper, conforme descrito em Wood *et al. (*2009), é um sistema que monitora MVs com o objetivo de detectar gargalos (*hotspots*) nos servidores físicos e atua de forma a mitigá-los. Tais gargalos são definidos como indisponibilidades de recursos em servidores físicos, que podem afetar o serviço das MVs que compartilham estes servidores.

Segundo Wood *et al.*(2009), os resultados apresentados pela proposta demonstram que *hotspots* singulares conseguem ser detectados e eliminados em menos de 20 segundos e que a proposta pode ser estendida para ambientes de grandes datacenters. Além disso, a proposta sugere a aplicação de duas abordagens.

A primeira é a abordagem caixa-preta (*black-box*), onde o monitoramento ocorre de forma independente do sistema operacional e das aplicações que são executadas nas máquinas virtuais. Deste modo, perfis de utilização de recursos são obtidos a partir do Monitor de Máquina Virtual (MMV) ou hipervisor de modo externo às MVs.

A segunda é a abordagem caixa-cinza (*gray-box*), que explora a execução dos processos e aplicações dos sistemas operacionais das MVs. Esta

abordagem utiliza-se de ferramentas de monitoramento instaladas em cada uma das MVs.

O sistema Sandpiper (WOOD et al., 2009) realiza o monitoramento de MVs na plataforma de virtualização Xen (BARHAM et al., 2003) e utiliza equações de predição para estimar pontos de estrangulamento na infraestrutura. Este sistema monitora os recursos CPU, memória e rede, tanto na abordagem caixa-preta quanto na abordagem caixa-cinza.

A diferença entre as abordagens envolve a forma como são obtidas as informações a respeito a utilização dos recursos. No caso da abordagem caixa-cinza, a informação é obtida a partir do sistema operacional instalado na MV. Na abordagem caixa-preta, os valores são obtidos a partir do MMV que possui uma visão externa das MVs.

Ao detectar *hotspots*, o Sandpiper aplica um algoritmo iterativo que ordena todos os servidores em função do seu volume (WOOD et al., 2009). O valor do volume é definido pela seguinte fórmula:

$$Vol = \frac{1}{1 - CPU} * \frac{1}{1 - memória} * \frac{1}{1 - rede} \quad (3.1)$$

O valor obtido por meio desta fórmula oscila entre 0 e 1 e é gerado pelo sistema para indicar o volume de recursos utilizados pelas MVs em função do uso do processador (CPU), da memória e da rede da MF.

Os parâmetros CPU, memória e rede correspondem à utilização destes recursos normalizada pelo número de processadores existentes, rede e memória disponível na MF.

Em seguida, o algoritmo ordena, dentro de uma mesma MF, os elementos virtuais que consomem mais recursos. Com base nestas informações, o sistema aloca iterativamente as MVs pertencentes à MF com maior volume na MF com menor volume, até que os *hotspots* sejam mitigados.

3.2 VOLTAIC

O sistema VOLTAIC (*Volume Optimization Layer To AssIgn Cloud resources*) é um gerente autônomo de recursos de ambientes de computação

em nuvem, que trata a alocação dos elementos virtuais, visando aumento da qualidade do serviço oferecido aos clientes ao evitar o desperdício de recursos computacionais (CARVALHO, 2012). O sistema utiliza a libvirt (REDHAT, 2013), que é uma biblioteca que permite a interação com múltiplas plataformas de virtualização para gerenciar servidores físicos.

O sistema utiliza controladores baseados em lógica nebulosa para detectar a saturação de máquinas físicas e propõe algoritmos de realocação automática que levam em consideração o perfil de utilização de cada MV e o perfil de recursos oferecidos por cada MF, tentando garantir que uma MV seja alocada na MF que forneça os recursos necessários de forma mais efetiva. Além disso, o VOLTAIC é compatível com quaisquer plataformas de virtualização que suportem a libvirt, tais como: Xen (BARHAM et al., 2003), VMWare (BUGNION et al., 2012), e KVM (KIVITY et al., 2007).

Ao adotar a utilização de uma interface única para gerenciar plataformas virtuais, amplia-se a aplicabilidade do sistema proposto, mas isso acarreta algum prejuízo quanto ao tipo de dado que pode ser extraído pela plataforma devido a limitações da libvirt (CARVALHO, 2012).

Segundo Carvalho (2012), apesar do problema citado, os resultados demonstram que a falta de algumas informações específicas das plataformas de virtualização não oneram sobremaneira o desempenho da proposta. O uso do VOLTAIC pelas plataformas está restrito àquelas que dispõem de mecanismos de migração em tempo real (*live migration*) de MVs. Além disso, é importante destacar que o VOLTAIC não considera o recurso rede na sua estratégia de alocação de elementos virtuais.

3.3 OUTROS TRABALHOS RELACIONADOS

Em Ando et al. (2009) é apresentado um sistema de proteção dinâmica de servidores Web que utiliza a técnica de migração em tempo real de MVs para tornar o serviço Web resiliente (resistente) a ataques DoS (*Denial-of-Service*). Este mecanismo utiliza o MMV para salvar, suspender e mover a MV que está sob ataque.

Um modelo eficaz de alocação de recursos dinâmicos para lidar com *flash crowd,* é apresentado por (CHANDRA; SHENOY, 2003). Este protótipo implementa uma arquitetura de servidores multicamada juntamente com mecanismos para monitoramento, detecção de carga, alocação dinâmica e balanceamento de carga. Nesta arquitetura, são usados servidores de reserva para reposição e suporte a sobrecarga, podendo até serem alocados vários servidores simultaneamente, dependendo da demanda.

Há um estudo proposto por Magalhães (*et al.*, 2011) que elenca algumas vantagens da utilização da migração de MVs como balanceamento de carga, tolerância a falhas e economia de energia. Estes autores apresentam as seguintes abordagens de migração de MVs: *non-live migration* ou *pure stop-and-copy e live migration*. Foi também evidenciado que a abordagem *stop-and-copy* possui vantagens em termos de simplicidade, e reduz o tempo total de migração em relação à abordagem *live migration*. Em contrapartida, o *downtime* sofrido pelas aplicações que são executadas na MV é maior, o que pode afetar de forma mais significativa os serviços disponibilizados por essas máquinas.

Em Nuaimi et.al (2012), Sidhu e Kinger (2013) e Kansal e Chana (2012) são apresentados estudos comparativos dos atuais de algoritmos de balanceamento da carga utilizáveis em ambientes de computação em nuvem. Nestes documentos, além uma visão geral, os algoritmos são categorizados e comparados com base nas suas propriedades. Além disso, os algoritmos também são comparados em razão da utilização de métricas como: desempenho, tempo de resposta, sobrecarga de rede, tempo necessário para migração, escalabilidade e tolerância à falhas.

Dos diversos algoritmos apresentados, dois algoritmos se destacaram e inspiraram o algoritmo de balanceamento de carga proposto neste trabalho: Compare and Balance descrito por Zhao e Huang (2009) e *Central Load For Virtual Machine Police* (CLBVM) descrito por Bhadani e Chaudhary (2010). O primeiro trata o problema de balanceamento de carga entre hosts físicos por meio migração adaptável de máquinas virtuais em tempo de execução. Um modelo de balanceamento de carga é projetado e implementado para reduzir o tempo de migração máquinas virtuais hospedadas na nuvem. O seu propósito é balancear a carga entre os servidores de acordo com uso do processador,

mantendo as MVs ativas. O algoritmo COMPARE and BALANCE também se propõe a realizar o balanceamento de carga baseado em amostragem e atinge o equilíbrio de modo bastante rápido. Este algoritmo assegura que a migração de MVs sejam sempre realizadas a partir de hosts físicos mais carregados para hosts menos carregados, porém tem como premissa que cada MF tem memória suficiente, fato que nem sempre é verdadeiro.

O Segundo algoritmo, CLBVM, propoe que o balanceamento de carga em ambiente de computação em nuvem seja feito de forma centralizada baseada no conhecimento de informações globais do cluster e na definição de uma política de balanceamento de carga. A política CLBVM considera os seguintes aspectos: que a carga da rede é constante e não ocorre grande variações; possui diferentes identificações; o processo coletor de carga roda continuamente em cada MF coletando um agregado a carga da CPU dos sistemas convidados (inquilinos); baseado nestas medições cada MF auto classifica sua carga como leve, moderada ou pesada; mensagens são trocadas entre o nó central e os demais nós; são priorizados os balanceamentos entre os levemente e os que estão com a carga pesada.

De acordo com Bhadani e Chaudhary (2010), experimentos realizados com CLBVM revelaram que a migração do sistema operacional adiciona um custo para o sistema global, que é minimizado quando é selecionando MV menos ativa. Foi verificado também que há necessidade de um servidor agindo como um backup em caso de falha do servidor mestre.

A análise dos trabalhos relacionados proporcionou o levantamento dos métodos, ferramentas, vantagens e desvantagens das diversas abordagens de balanceamento de carga e tratamento de sobrecarga em ambiente de computação em nuvem. Com base neste levantamento, foram estabelecidos os requisitos, funcionalidades e o ambiente computacional necessário à experimentação da arquitetura proposta.

4 DESCRIÇÃO DA ARQUITETURA PHOENIX

Este capítulo apresenta a arquitetura proposta para este trabalho, chamada de Phoenix.

A Phoenix é um sistema de tratamento de sobrecargas transientes em ambiente de computação em nuvem. Este sistema utiliza gerência automática de MFs e MVs e é aplicável ao modelo de Infraestrutura como Serviço (IaaS) descrito por Mell e Grance (2012).

No ambiente de computação em nuvem, existem grandes conjuntos de MFs e de MVs. Mecanismos como escalonamento vertical e horizontal descritos por Mei et al.*(*2008) garantem o comportamento elástico neste ambiente, porém, isoladamente não são capazes de conter eventuais sobrecargas transientes como *flash crowds* que podem sobrecarregar os recursos computacionais existentes, podendo causar inclusive a quebra de ANSs por indisponibilidade dos serviços hospedados. Neste trabalho, os ANSs estabelecidos entre os clientes e o fornecedor são utilizados para configuração das máquinas virtuais deste modo, cada máquina virtual configurada representa um ANS estabelecido.

Cada MV pode apresentar um padrão diferente de utilização de recursos (CARVALHO, 2012). Desta forma, além da alocação correta de MVs nas MFs disponíveis, é fundamental a associação destas com estratégias que possam suportar eventuais sobrecargas para garantir o atendimento dos ANSs de cada MV ou serviço de infraestrutura hospedado.

Além do tratamento reativo da sobrecarga em MFs ou em MVs a arquitetura se predispõe a realizar o balancemanto de carga do cluster de modo preventivo, possibilitando que as MFs que suportam os serviços trabalhem com suas cargas equilibradas. Isto evita que em condições normais algumas MFs trabalhem sobrecarregas, correndo o risco de terem seu desempenho e seus serviços comprometidos, enquanto outras MFs trabalhem subutilizadas. Neste contexto, surge a proposta da arquitetura Phoenix.

Com relação a abordagem estabelecida para aquitetura Phoenix, optou-se pela abordagem caixa-preta, por esta permitir que decisões de migração de MVs entre as MFs da nuvem pudessem ser tomadas de forma não intrusiva, ou seja, sem qualquer interferência na configuração do sistema operacional ou dos

aplicativos residentes em cada MV. Este é um modo bastante adequado de verificação quando se oferece infraestrutura como serviço (IaaS – *Infraestructure as a Service*), uma vez que o provedor de serviços não deve interferir internamente nas MVs contratadas.

A arquitetura Phoenix, utiliza um mecanismo para fazer a escolha da estratégia de migração a ser empreendida que se baseia na situação atual das máquinas do cluster e dos limites estabelecidos pela faixa de equilibrio.

Quando não estiver tratando problemas de sobrecarga de recursos e se for necessário, a arquitetura deve realizar o balanceamento de carga de forma proativa em relação à possibilidade de sobrecarga de MFs por meio do remanejamento de MVs (balanceamento de carga).

Visando o menor *downtime* dos serviços, a arquitetura proposta considera a opção de migração em tempo real (*live migration*) para migração de MVs.

Por questão de economia de recursos, não foi considerada a possibilidade de uma reserva de recursos computacionais que ficariam a disposição em caso de uma demanda além do estabelecindo nos ANSs.

Por fim, devido às vantagens do uso da libvirt, descritas por Carvalho (2012), decidiu-se utilizar esta biblioteca para obtenção de informações do ambiente e das MFs e MVs de modo a subsidiar a tomada de decisão e permitir o uso de vários monitores de máquinas virtuais.

O nome Phoenix, atribuído à arquitetura, é uma alusão ao pássaro da mitologia grega, que morria, mas depois de algum tempo renascia das próprias cinzas. O pássaro chamado fênix tornou-se um símbolo de imortalidade, de renascimento e de força. Da mesma forma que este pássaro da mitologia grega renasce das próprias cinzas, a arquitetura Phoenix se reestrutura a cada evento relevante, visando o balanceamento de carga entre as MFs do cluster ou o suporte às sobrecargas em MFs ou MVs, propiciando assim um novo arranjo de MFs e MVs hospedadas em ambientes de computação em nuvem, suportando assim, os efeitos causados por eventuais sobrecargas transientes.

4.1 REQUISITOS DA ARQUITETURA PHOENIX

A Phoenix deve tratar eventos de sobrecarga observando os níveis de serviços contratados agindo por meio de uma administração eficaz da elasticidade do ambinente e das capacidades das MVs e das MFs considerando os seguintes requisitos:

- Migrar MVs automaticamente - este é um requisito necessário para manutenção de um ambiente virtualizado de acordo com níveis de serviço acordados, uma vez que se exige habilidade e agilidade para responder às mudanças súbitas de carga de trabalho. Segundo Wood et al. *(*2009), a migração de MV iniciada manualmente não tem esta agilidade e também é mais propensa a erros, pois cada reconfiguração pode exigir migrações ou trocas de vários servidores virtuais para reequilibrar a carga do sistema.

- Alocar e gerenciar os recursos automaticamente - cada MV pode ser configurada para designar um percentual de recursos físicos diferente a cada consumidor. Segundo Franke (2010), à medida que estes recursos virtuais extrapolam ou põem em risco a capacidade da MF, as MVs devem ser realocadas.

- Monitorar os recursos e requerimentos de capacidade e depois balancear e configurar as MVs em tempo real - este requisito se faz necessário para atender o aumento e a diminuição da demanda nos diferentes níveis de requerimento de serviços acordados para os diferentes clientes. Para tanto, novos mecanismos e algoritmos devem ser desenvolvidos (FRANKE, 2010).

- Rastrear métricas de desempenho, máximas, mínimas e detectar e informar anomalias - este requisito estabelece que ferramentas também precisam possibilitar que serviços confiáveis de auditoria possam monitorar externamente o ambiente, visando o atendimento dos ANSs. Em Franke (2010), foi destacado que quando anomalias

são detectadas, o sistema deve ser capaz de resolver os problemas automaticamente. Em casos onde o sistema não consegue resolver o problema sozinho, deve disparar um alarme apropriado para que as medidas necessárias sejam tomadas.

- Tratar mudanças súbitas da carga de trabalho tendo como referência os ANSs - este requisito estabelece que a arquitetura deve identificar mudanças súbitas da carga de trabalho de modo que as MFs tenham condições de disponibilizar os recursos computacionais necessários às MVs, tendo como referência os ANSs estabelecidos (FRANKE, 2010). Este requisito é fundamental para Phoenix, pois o ANS é o grande orientador das ações da arquitetura

A partir destes requisitos, foram estabelecidas as principais funcionalidades da arquitetura.

4.2 VISÃO GERAL DA ARQUITETURA PHOENIX

Para cumprir os objetivos propostos, observando os requisitos estabelecidos na seção 4.1, foi construida a arquitetura Phoenix, que suporta sobrecargas momentâneas em ambiente de computação em nuvem por meio das seguintes atividades: automatização das tarefas de monitoramento da utilização de recursos de sistema, gerenciamento da hospedagem das MVs, detecção e suporte a sobrecargas momentâneas, tendo como referência os níveis de serviço acordados e balanceamento de carga entre as MFs.

A arquitetura Phoenix é composta pelos módulos: Monitor, Analisador e Migrador e pode ser esquematicamente visualizada na FIG. 4.1.

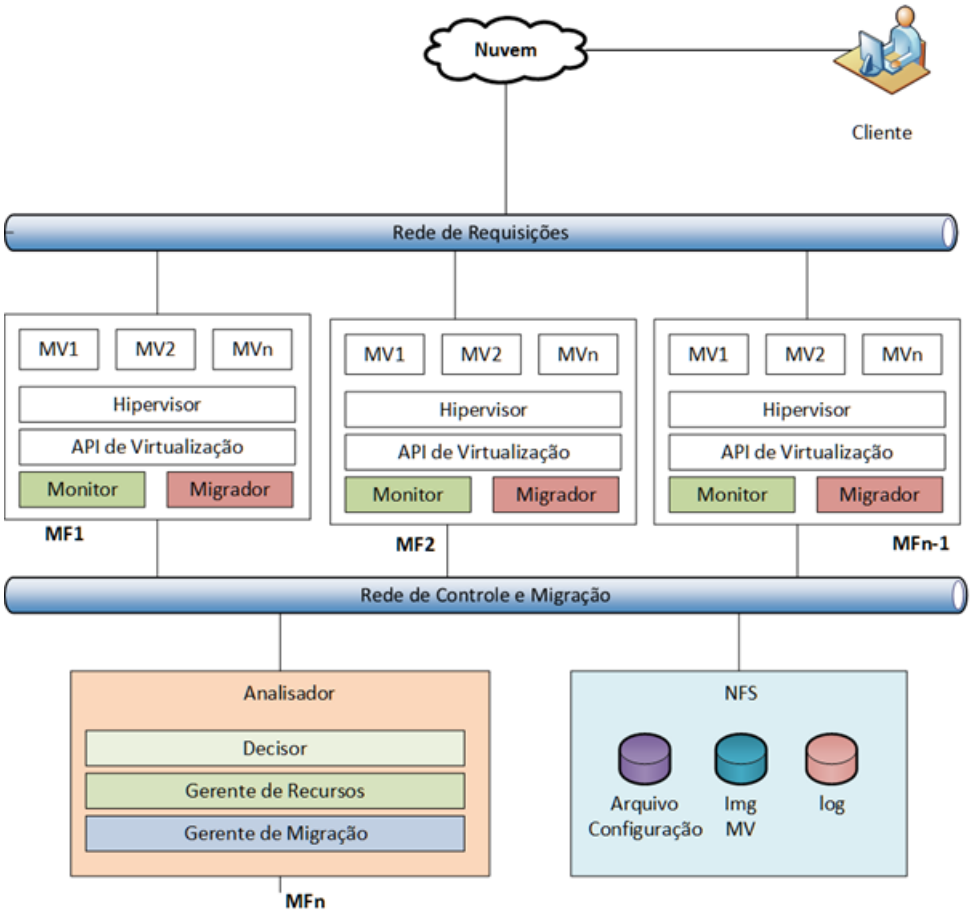

FIG. 4.1 Arquitetura Phoenix

4.2.1 DESCRIÇÃO DOS COMPONENTES DA ARQUITETURA PROPOSTA

Os elementos propostos na arquitetura Phoenix foram organizados de modo a atender os objetivos deste trabalho e estão destacados em cores, quais sejam: monitor, migrador, analisador, repositório NFS e redes distintas para requisições e controle.

A seguir são descritos os diversos componentes da arquitetura Phoenix:

MF_i – representam as MFs existentes na arquitetura. Estas MFs hospedam os serviços oferecidos pela nuvem. As máquinas físicas MF_1 a MF_{n-1} formam o cluster de máquinas que hospedam os serviços de IaaS e tem os seus recursos gerenciados pela arquitetura. Enquanto a MF_n hospeda o módulo Analisador a demais MFs hospedam os módulos Monitor e Configurador.

MV_i – são instâncias das máquinas virtuais, configuradas conforme os requisitos estabelecidos nos ANSs para IaaS gerenciadas pela arquitetura. Nestas MVs são executados os sistemas operacionais (SOs) e as aplicações dos clientes do serviço de infraestrutura.

Rede de Requisições – corresponde a infraestrutura de rede por onde são feitas as requisições dos usuários aos serviços hospedados nas MVs.

Rede de Controle e Migração – corresponde a infraestrutura de rede por onde ocorre a troca de mensagens entre as MFs e por onde são feitas as migrações das MVs, sendo esta de uso exclusivo para os serviços de gerenciamento do Cluster.

Módulo Monitor - é responsável por colher informações de utilização dos recursos (CPU, memória e rede) das MVs e MFs e gerar valores médios de utilização desses recursos. Ao confrontar estes valores com limiares configuráveis, identifica eventos relevantes como: sobrecarga da MF, sobrecarga de MVs, variação de utilização dos recursos CPU, memória ou rede das MFs acima ou abaixo do limiar configurável, repassando-os ao módulo Analisador.

Módulo Analisador – analisa eventos relevantes provenientes do módulo Monitor e avalia a necessidade de movimentação das MVs, seja por sobrecarga de recursos da MF (CPU, memória e/ou rede), sobrecarga do recurso de rede disponibilizado para cada MV, necessidade de recursos definidos para cada MV, conforme descrito no respectivo ANS, ou a necessidade de equilibrar a carga entre as diversas MFs do cluster. Uma vez definido o evento relevante a ser tratado, são estabelecidas estratégias que, quando envolvem a movimentação de MVs, são encaminhadas ao módulo Configurador.

Hipervisor – é o Monitor de Máquinas Virtuais (MMV), responsável por implementar e gerenciar as MVs. Recebe orientações provenientes do módulo Configurador.

API de Virtualização – é fornecida pela biblioteca libvirt. Essa API está relacionada aos recursos de virtualização do Linux que suporta diversos hipervisores, incluindo o KVM, que foi escolhido como plataforma de virtualização para implementação da arquitetura proposta.

Módulo Configurador - transforma as estratégias de migração definidas pelo módulo Analisador em comandos executáveis pelos MMVs, determina sua execução e informa a sua finalização ao módulo Analisador.

Arquivos de Log – local onde estão armazenadas as informações do histórico sobre utilização de recursos por parte das MFs e MVs, eventos relevantes identificados pelos monitores, e ações planejadas pelo analisador e executadas pelo Configurador. Estas informações são registradas por cada módulo da arquitetura no seu respectivo Log.

Arquivo de Configuração – arquivo onde estão armazenadas as informações dos acordos de nível de serviço e informações de configuração das MFs e MVs, bem como dos limites usados pela arquitetura. Estas informações podem ser consultadas pela arquitetura quando necessário.

Como a arquitetura adota a abordagem de monitoramento Black-Box, vale destacar que o monitoramento é feito de forma externa às MVs. Um outro aspecto a ser observado, é que a arquitetura prevê o uso de uma API de virtualização para a comunicação com os MMVs. Isto possibilita que a arquitetura utilize diferentes MMVs.

4.2.2 FUNCIONAMENTO DA ARQUITETURA PROPOSTA

O funcionamento da Phoenix foi inspirado no ciclo OODA descrito por Boyd (1981), em virtude da forte semelhança entre as fases do ciclo e os processos observados em uma arquitetura para tratamento de sobrecargas em máquinas virtuais, quais sejam: observação da situação das máquinas, orientação das causas da sobrecarga, planejamento e decisão sobre quais ações a serem tomadas e finalmente execução das migrações necessárias. Este comportamento é executado em ciclos similares ao OODA.

Assim como nas fases do ciclo OODA, o comportamento dinâmico da arquitetura Phoenix pode esquematicamente ser representado, conforme a FIG 4.3 e descrito nas próximas subseções.

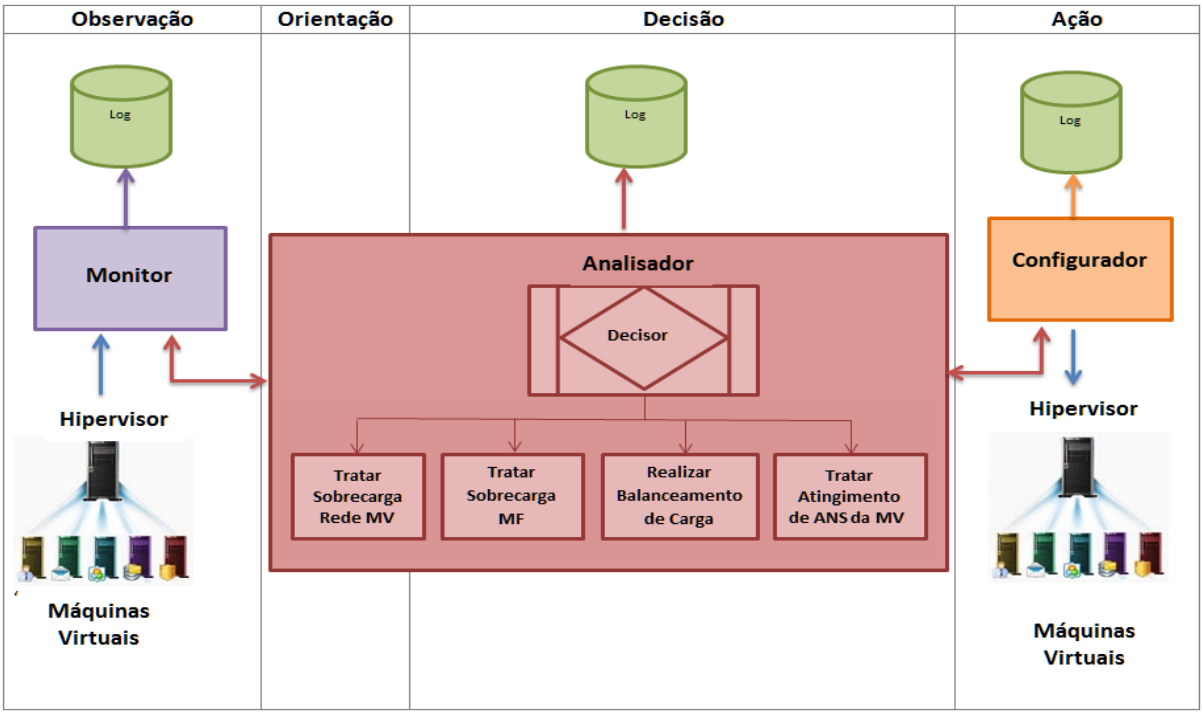

FIG 4.3 Funcionamento da Arquitetura Phoenix

4.2.2.1 MÓDULO MONITOR

Este módulo corresponde à fase de Observação do ciclo OODA e tem por finalidade colher os perfis de utilização de recursos (CPU, memória e rede) das MVs e MFs, identificar eventos relevantes de acordo com limiares configuráveis e enviar mensagens contendo estes perfis e a indicação da ocorrência destes eventos ao módulo Analisador.

Somente são enviadas mensagens ao módulo Analisador quando houver uma alteração relevante do estado das MVs ou MFs monitoradas ou ao término de cada migração.

Uma alteração relevante corresponde à sobrecarga de MFs, MVs ou uma variação acima de um determinado limite configurável da utilização da CPU, memória e/ou rede das MFs e/ou MVs. Assim, caso as MVs ou MFs permaneçam em um estado estável, ou seja, dentro do limite configurável de utilização dos recursos, nenhuma mensagem é enviada ao módulo Analisador, minimizando a troca de mensagens entre os elementos da arquitetura.

Ao coletar os perfis de utilização dos recursos da MFs e MVs, o Monitor verifica a ocorrência de sobrecarga utilizando o método proposto por Ferreira e Martins (2013), modificado para também identificar sobrecargas em MVs. Os valores de utilização dos recursos obtidos por este método são médios. Isto permite que a comparação entre os valores obtidos e os limites configuráveis de sobrecarga seja feita considerando valores médios acumulados, evitando que valores de pico de uso dos recursos sejam considerados sobrecargas.

O algoritmo do módulo Monitor lê os perfis coletados e calcula o percentual acumulado de utilização de cada um dos recursos (CPU, memória e rede), utilizando na fórmula (4.1) cujo os componentes são descritos na TAB 4.1.

$$N_{i+1} = \mu * N_{instantâneo} + (1 - \mu) * N_i \qquad (4.1)$$

TAB. 4.1 Elementos da Fórmula para Cálculo Média de Utilização de Recursos

Elemento	Descrição
N_{i+1}	É uma previsão para o próximo valor instantâneo a ser capturado.
μ	É uma constante com valor expresso entre 0 e 1 (0 < μ < 1), que expressa o quanto o valor instantâneo que será levado em consideração para o cálculo da média. Serve como um redutor que permitirá um retardo maior ou menor na detecção de uma sobrecarga.
$N_{instantâneo}$	É o valor capturado de uso do recurso em porcentagem na iteração *i*.
N_i	É o valor acumulado de uso do recurso da iteração anterior.

Pode-se considerar que o N_i representa o histórico dos valores calculados, $N_{instantâneo}$ é o valor atual e N_{i+1} é uma previsão para o próximo valor instantâneo a ser capturado, utilizando-se de uma média móvel exponencial, conforme descrito em Silberschatz et al.(2005) apud Ferreira e Martins (2013) para o cálculo da previsão da duração do próximo *burst* de CPU no algoritmo de escalonamento SJF (*Shortest Job First*). Um método semelhante é descrito por WOOD et al.(2009).

O método proposto impede que sejam geradas mensagens ao analisador de situações de pico temporárias, que efetivamente não caracterizam o alcance dos limites de sobrecarga estabelecidos.

A fórmula descrita na expressão (4.1) é aplicada a cada um dos recursos (CPU, memória e rede) monitorados, de modo que seja possível a determinação do grau de utilização desses recursos, sem a contaminação gerada por uma situação de pico.

Os valores dos percentuais de utilização de recursos das MVs e/ou MFs obtidos a partir deste método são comparados com os percentuais acumulados e confrontados com os limites de sobrecarga das MFs e MVs, seguindo determinada ordem de prioridade, conforme descrição a seguir:

1. Após a leitura dos percentuais de utilização de recursos é verificado se houve a superação do limiar definido para utilização da rede das MVs. Caso tenha ocorrido, é gerada e encaminhada uma mensagem de “sobrecarga de rede da MV” para o Analisador.
2. Caso o evento descrito anteriormente não tenha sido identificado, é verificado se houve superação dos limites percentuais definidos para os recursos (CPU, memória ou rede) das MFs. Se houver a superação do limiar de qualquer destes recursos, será enviada uma mensagem de “sobrecarga na MF” para o Analisador.
3. Caso os limiares definidos para sobrecarga de MFs não tenham sido atingidos, os percentuais são comparados com percentuais acumulados, obtidos das leituras anteriores, de modo que seja verificado se há uma variação relevante destes valores. Se isto ocorrer, é enviada uma mensagem de “evento relevante” para o Analisador. A variação percentual que define um evento relevante é definida pelo administrador da arquitetura.
4. Caso não tenha ocorrido nenhum dos eventos anteriores descritos anteriormente, é verificado se houve a superação dos limites correspondentes aos demais recursos das MVs (CPU, memória). Caso tenha ocorrido a superação do limiar de qualquer destes recursos será

enviada uma mensagem de “sobrecarga de CPU ou Memória da MV” para o Analisador.

Ao identificar os eventos nesta ordem, o módulo Monitor estabelece uma prioridade para envio de mensagens ao Analisador, classificando os eventos contidos nestas mensagens por ordem impacto ou relevância. Estes são tratados pelo módulo Analisador, obedecendo esta prioridade.

Para exemplificar esta priorização, pode-se citar o caso em que o Monitor identifique que houve uma sobrecarga de requisições na rede da MV ou mesmo um *flash crowd* em um serviço hospedado em uma MV e além disso, também esteja ocorrendo uma sobrecarga do recurso CPU da MV.

A sobrecarga na rede da MV, por ser um evento cuja demanda não pode ser contida por meio configurações estabelecidas pelo MMV, pode impedir que a MF mantenha o ANS dos demais serviços hospedados, enquanto que a sobrecarga da CPU da MV pode ser contida pela configuração estabelecida pelo MMV. Sendo assim, é atribuído à sobrecarga de rede maior prioridade que o atingimento do limiar estabelecido para utilização da CPU da MV. Com base neste raciocínio foi estabelecida a TAB. 4.2 que estabelece a prioridade dos eventos relevantes identificados pelo módulo Monitor.

TAB. 4.2 Prioridade dos Eventos para o Envio de Mensagens do Monitor

Prioridade	**Rede da MV Sobrecarregada**	**MF Sobrecarregada (CPU, Memória e/ou rede)**	**Evento Relevante**	**Atingimento de ANS de MV**
1	SIM	Qualquer Combinação de Eventos		
2	NÃO	SIM	Qualquer Combinação de Eventos	
3	NÃO	NÃO	SIM	Qualquer Combinação de Eventos
4	NÃO	NÃO	NÃO	SIM

Como exemplo da utilização desta tabela, pode-se fazer a interpretação da linha 4 da TAB. 4.2 que expressa que o Monitor detectou que não há indícios de sobrecarga do recurso rede de nenhuma das MVs (coluna Rede da MF Sobrecarregada = “NÃO”), que a MF não está sobrecarregada (coluna MF Sobrecarregada = “NÃO”), que o cluster está balanceado (coluna Cluster Desbalanceada = “NÃO”) e que foram atingidos os limites de ANS para CPU, memória ou rede de alguma MV contida na MF monitorada (coluna Atingimento de ANS de MV = “SIM”). Ela também mostra que, neste caso, deverá ser tomada a ação “Tratar Atingimento Limite de ANS” referente à MV.

Também pode-se observar o estabelecimento da priorização das ações a serem empreendidas, pela numeração estabelecida na coluna Prioridade da TAB. 4.2, nesta é identificado, por exemplo, que a sobrecarga da rede da MV tem a maior prioridade, seguida pelo tratamento de sobrecarga da MF, balanceamento do cluster e, por fim, o atingimento do ANS das MVs.

A TAB. 4.2 apresenta todas as combinações de eventos relevantes possíveis para os três recursos envolvidos (CPU, memoria e rede), sendo que foram eliminadas da tabela as combinações de eventos repetidas. Deste modo, uma vez detectados e priorizados os eventos são enviados por meio de mensagens ao Analisador. As mensagens enviadas geram diferentes ações a serem tomadas pelo Decisor.

O algoritmo, descrito na FIG. 4.2, mostra as principais funcionalidades do módulo Monitor.

```
Input :  StopUpdate/StartUpdate
Output: Evento_Relevante (Perfis MV,MF)

1 Faça enquanto paraliza_Monitor = 'não'
2       obterMFInfo()
3       mfCPU=calcula_mf_CPU_Acumulado(MF)
4       mfMEM=calcula_mf_MEM_Acumulado(MF)
5       mfREDE= calcula_mf_REDE_Acumulado(MF)
6       lista_MV=obterMVInfo()
7       Para cada MV da lista_MV
8           mvCPU=calcula_mv_CPU_Acumulado(MV)
9           mvMEM=calcula_mv_MEM_Acumulado(MV)
10          mvREDE= calcula_mv_REDE_Acumulado(MV)
11          Se (mv_REDE > LIMIT_MV_REDE)
12              informa_sobrecarga_REDE_MV(MV)
13          Senão  SE(mfCPU > LIMIT_MF_CPU ou mfMEM > LIMIT_MF_MEM ou mfREDE > LIMIT_MF_REDE)
14              informa_sobrecarga_MF()
15              Se(mvCPU > LIMIT_MV_CPU ou mvMEM > LIMIT_MV_MEM)
16                  informa_sobrecarga_CPU_MEM_MV()
17              Senão Se((mfCPU> cpuAnterior + ALTERACAO_RELEVANTE ou mfCPU < cpuAnterior - ALTERACAO_RELEVANTE) ou \
18                  (mfMEM > memAnterior + ALTERACAO_RELEVANTE ou mfMEM < memAnterior - ALTERACAO_RELEVANTE) ou \
19                  (mfREDE > redeAnterior + ALTERACAO_RELEVANTE ou mfREDE < redeAnterior - ALTERACAO_RELEVANTE))
20                  cpuAnterior = mfCPU
21                  memAnterior = mfMEM
22                  redeAnterior = mfREDE
23                  informa_Evento_Relevante()
24              Fim-se
25          Fim se
26      Fim para
27 Fim enquanto
```

FIG. 4.2 Algoritmo Principal do Módulo Monitor

Deste modo, somente em situações que exijam a atenção do Analisador ocorre a troca de mensagem entre os elementos da arquitetura. Esta medida reduz a possibilidade de excesso de troca de mensagens entre os Monitores e o Analisador, além de reduzir o processamento realizado no Analisador.

4.2.2.2 MÓDULO ANALISADOR

Este módulo corresponde às fases de Orientação e Decisão do ciclo OODA e é responsável por avaliar as mensagens provenientes dos vários Monitores, determinar a movimentação ou fixação das MVs na MF original, seja por

sobrecarga das MFs ou de rede das MVs, ou pela necessidade de equilibrar a carga das diversas MFs do cluster.

O módulo Analisador entra em ação a partir da chegada das mensagens provenientes dos módulos Monitores. É composto pelos submódulos: Gerente de Recursos, Decisor e Gerente de Migração.

O submódulo Gerente de Recursos recebe as mensagens enviadas pelos Monitores e atualiza a estrutura em memória que suporta as decisões do módulo Analisador. Este submódulo também calcula o volume e o custo de cada MV, a carga total de cada MF (CTMF) e a carga média do cluster (CMC). Estas informações e o conteúdo das mensagens enviadas pelo Monitor são repassadas ao submódulo Decisor.

O submódulo Decisor analisa as informações das MVs e MFs do cluster repassadas pelo submódulo Gerente de Recursos. O conteúdo de cada mensagem é verificado e submetido a um conjunto de regras descrito a seguir.

Se for confirmado o alcance dos limites de sobrecarga de MV ou MF ou a necessidade de balanceamento, são acionados os métodos que se encarregam de tratar o evento detectado. Os métodos são acionados de forma hierárquica refletindo a prioridade de tratamento dos eventos estabelecida pelo módulo Monitor ao selecionar as mensagens que foram enviadas. Os métodos são: tratar Sobrecarga da Rede da MV, Sobrecarga de MF, suportar Balanceamento de Carga e Tratar Atingimento de ANS.

A **Erro! Fonte de referência não encontrada.** apresenta as ações realizadas a partir das mensagens enviadas pelo Monitor. As mensagens dos Monitores possuem uma prioridade de tratamento embutida e acionam diferentes ações a serem tomadas pelo Decisor.

TAB. 4.3 Ações Realizadas pelo Módulo Analisador

Prioridade	Evento	Ação a ser empreendida pelo Analisador
1	Rede da MV Sobrecarregada	Tratar Sobrecarga da Rede da MV
2	MF Sobrecarregada (CPU, Memória e/ou rede	Tratar Sobrecarga de MF
3	Evento Relevante	Suportar Balanceamento de Carga
4	Atingimento de ANS de MV	Tratar Atingimento de ANS

Os eventos da TAB. 4.3 são descritos como se segue:

Rede MV Sobrecarregada - indica que o percentual de utilização da rede da MV é maior que o previsto em ANS. Portanto, a utilização do recurso rede por MV é maior que o limiar estabelecido, o que indica sobrecarga deste recurso, podendo ocasionar uma sobrecarga na rede da MF, dificultando ou mesmo impedindo o acesso das outras MVs ao recurso.

MF Sobrecarregada - indica que o percentual de utilização de recursos CPU, memória ou rede é maior que o limiar configurado, ou seja, a MF está sobrecarregada. Assim, devem ser disponibilizados os recursos necessários às MVs de modo que não haja quebra dos ANSs.

Evento Relevante – indica que houve uma variação significativa (configuraárvel) que pode indicar a necessidade uma ação subsequente. Neste caso faz-se necessário verificar a necessidade de balanceamento de carga. Este evento pode indicar que existe uma ou mais MFs com Carga Media Total da MF (CMTF) acima ou abaixo do limiar estabelecido para a Carga Média do Cluster (CMC). Se for verificada a necessidade de balanceamento, a arquitetura deverá balancear o cluster movimentando MVs entre as MF de modo equilibrar as CMTFs.

Atingimento de ANS de MV (CPU, memória ou rede) - indica se foi atingido o limite de utilização dos recursos CPU, memória ou rede acordados e configurados para cada MV. Possivelmente, a MV necessita de ajustes no seu ANS de modo a suportar a demanda imposta. Isto poderá ser avaliado estatisticamente, e em momento oportuno, subsidiar a sugestão de mudanças no ANS a serem feitas ao dono do Serviço. Não faz parte do escopo de atividades da arquitetura a alteração de ANSs. Neste caso simplemente é feito um registro do ocorrido.

Os tópicos seguintes descrevem como o algoritmo do módulo Analisador trata suas condições de acionamento:

Ação Tratar Sobrecarga da Rede da MV - O algoritmo do módulo Analisador estabelece que é necessário tratar Sobrecarga da Rede da MV quando o Monitor informa por meio de uma mensagem (OVERNETMV) a ocorrência deste evento. Isto significa que o percentual de uso da rede de alguma MV ultrapassou o limiar estabelecido conforme o ANS. Assim, se, por exemplo, um ANS estabelece que uma MV pode utilizar até 100 Mbps e este limite for ultrapassado (considerando a obtenção dos percentuais de utilização dos recursos conforme o descrito na Seção 4.2.2.1), então algoritmo analisa as possibilidades de movimentação de MVs, determina a movimentação proativa das MVs não afetadas e o isolamento da MV com sobrecarga do recurso rede. Esta medida tem por objetivo minimizar os efeitos que o aumento súbito da demanda por este recurso pode causar no funcionamento da MF e evitar comprometimento do ANS das demais MVs hospedadas na mesma MF.

O tratamento de sobrecarga de rede torna-se necessário porque este recurso, em alguns hipervisores atuais como o KVM (KIVITY et al., 2007) e o Xen (BARHAM et al., 2003), diferentemente da CPU e da memória, não pode ser limitado para cada MV de uma MF, na prática, as tratativas das requisições externas realizadas aos serviços hospedados pelas MV não se limitam ao limiar imposto pelo hipervisor. Além da utilização de recursos prevista para os serviços hospedados, eventuais excessos também são tratados pela MF que suporta o serviço. A MF pode ficar sobrecarregada se a demanda for excessiva. Neste caso, eventuais sobrecargas da MF podem afetar os demais serviços ocasionando a quebra de ANS. Assim, este recurso precisa ser gerenciado de

modo que eventuais sobrecargas sejam contidas evitando o comprometimento dos demais serviços de IaaS.

Para realizar o tratamento de sobrecarga da rede da MV, a MV afetada é fixada na MF origem e as outras MVs que estão hospedadas na mesma MF são migradas, enquanto houver aumento da utilização do recurso rede da MV em relação ao acordado no ANS. Como resultado, a MF que contém a MV afetada pode dedicar grande parte do seu poder computacional exclusivamente para atender as requisições oriundas das requisições excedentes (não contempladas no respectivo ANS), preservando o ANS das demais MVs.

As MVs não afetadas devem ser retiradas até que permaneça na MF somente a MV afetada ou que seja cessada a sobrecarga da rede da MV afetada.

Em relação à seleção das MVs a serem migradas, é utilizada a ordenação crescente das MVs por custo. Assim, pretende-se liberar recursos o mais rápido possível. O conceito de custo está associado ao conceito de volume de uma MV, definido em Wood *et al.*(2009).

Os parâmetros CPU, memória e rede correspondem à utilização destes recursos normalizada pelo número de processadores existentes, uso de rede e memória disponível na MV. Para o cálculo do custo de migração é utilizada uma adaptação da função de Cobb-Douglas, conforme o descrito em Ferreira e Martins (2013):

$$Custo = Vol^{\alpha} * Img^{\beta} \qquad (4.2)$$

Onde Vol é o volume da máquina virtual, Img é o tamanho da imagem da memória da MV que deverá ser migrada e α, β são constantes onde $0 < \alpha, \beta < 1$ e $\alpha + \beta = 1$. Este valor representa uma fotografia momentânea da utilização de recursos (CPU, Memória e rede) pela MV. Quanto maior o valor do custo, mais tempo será gasto para migração da MV.

Em relação à seleção das MFs candidatas a receber a MV selecionada, realiza-se a ordenação crescente das MFs candidatas, priorizando as MFs menos carregadas considerando a carga média do cluster (CMC). Deste modo,

irão receber as MVs aquelas que tiverem o maior afastamento em relação à CMC e tenham condições de absorver a carga da MV.

Para receber a MV, deverá ser escolhida a MF com Carga Total (CTMF) que possui o maior distanciamento da MF em relação à CMC que, em termos absolutos, corresponde ao menor valor de CTMF. Este procedimento visa manter o cluster balanceado e reduzir o número de migrações.

A carga Média do Cluster representa a média dos valores reais de CPU, memória e rede extraídos após a ponderação feita com pesos atribuídos pelo administrador, totalizados por MF e divididos pelo número de MFs. Assim, Para efetuar o cálculo da Carga Total da MF (CTMF), o algoritmo lê os perfis de recursos (CPU, memória e rede) utilizados, multiplica estes valores por pesos configuráveis estabelecidos pelo administrador e executa a soma destes valores obtendo assim CTMF

Os pesos representam o grau de importância que o administrador do cluster atribui a cada recurso utilizado pelas MF e pode interferir nas tratativas dos eventos relevantes pelo módulo Analisador.

O cálculo da CTMF pode ser sintetizado utilizando na fórmula a seguir, cujas variáveis são descritas na TAB. 4.4:

$$\mathrm{CTMF} = \left(\frac{(\mathrm{CPU} * \mathrm{Pcpu}) + (\mathrm{MEM} * \mathrm{Pmem}) + (REDE * Prede)}{(Pcpu + Pmem + Prede)} \right) \qquad (4.3)$$

Para calcular a Carga Média do Cluster (CMC), o algoritmo executa o somatório das Cargas Totais das Máquinas Físicas (CTMFs), e em seguida divide pela quantidade total de MFs.

TAB. 4.4 Elementos da Fórmula para Cálculo Carga Média do Cluster

CPU	É o percentual de recurso de CPU pela MF
MEM	É o percentual de recurso de MEM pela MF
REDE	É o percentual de recurso de REDE pela MF
Pcpu	É o valor do peso aplicado ao recurso CPU
Pmem	É o valor do peso aplicado ao recurso MEM
Prede	É o valor do peso aplicado ao recurso REDE
S	Representa cada MFs do cluster
m	Representa o número total de MFs do cluster.

A FIG. 4.3 descreve as principais funcionalidades do algoritmo Tratar Sobrecarga de MV.

```
Input :  MF Origem
Output: Dados_de_Migração

// Trata Sobrecarga de Rede de MV
Se mensagem_MF_origem  == "OVERNETVM"
        Se MF_origem.estado == "MIGRANDO"
                msg ('MF de origem não pode ser tratada pois está migrando alguma MV.')
                retorne
        end
        fixar_MVs_sobrecarregadas(MF_origem)
         Se MF_origem possui MV_não_ sobrecarregada
                MV_escolhida = escolher_MV_menor_custo( MFOrigem)
                MF_destino = escolher_MFDestino()
                Enviar_Dados_Migracao(MV_escolhida, MF_origem, MF_destino)
        end
end
```

FIG. 4.3 Algoritmo Tratar Sobrecarga de MV

Ação Tratar Sobrecarga de MF - o Analisador estabelece que é necessário tratar a sobrecarga de MF quando for alcançado o limiar estabelecido e configurável pelo administrador do cluster, para os recursos CPU, memória ou

rede da MF. Este percentual de uso deve ser um valor próximo e abaixo de 90% (valor que pode ser configurado) de modo a permitir que eventuais migrações possam ser realizadas sem prejuízo da utilização de recursos pelas MVs, conforme o estabelecido nos ANSs.

As estratégias para escolha das MVs a serem migradas e das MFs que receberão essas MVs são similares às estratégias utilizadas no tratamento de sobrecarga de rede de MV.

O algoritmo irá movimentar as MVs que não estão sob ameaça de sobrecarga de rede para outras MFs que tenham capacidade de recebê-las, priorizando as MVs que possuírem menor custo. Esta medida visa migrar prioritariamente MVs que estiverem utilizando menos recursos, de modo que a MF sobrecarregada tenha os recursos liberados no menor tempo possível em prol da manutenção dos ANSs e da melhoria das condições de funcionamento da MF original.

Em relação à seleção das MVs a serem migradas, é utilizada a ordenação crescente das MVs por custo utilizando o conceito de volume, já discutidos anteriormente.

Para receber a MV, deverá ser escolhida a MF com menor Carga Total (CTMF) que possa receber a MV sem que esta tenha seus recursos sobrecarregados, ou seja, não fiquem com percentuais de utilização dos recursos CPU, memória ou rede acima dos limiares estabelecidos, caso ocorra a migração. Este procedimento evita a sobrecarga da MF recebedora e possibilita manter o cluster balanceado após a migração.

A FIG. 4.4 apresenta o algoritmo que implementa a ação tratar sobrecarga de MF realizada pelo módulo Analisador.

```
Input :MF_Origem
Outpu Dados_de_Migração

1  // Trata Sobrecarga de MF
2  Se mensagem_MF_origem  == "OVERNETMF"
3      Se MF_origem.estado == "MIGRANDO"
4          msg ('MF de origem não pode ser tratada pois está migrando alguma MV.')
5          retorne
6      end
7      Identificar_MVs_disponíveis(MF_origem)
9          MV_escolhida = escolher_MV_menor_custo( MF_Origem)
           MFs_Disponível =Identificar_MFs_disponíveis(MFs_cluster)
10         MF_destino = escolher_MF_menorCTMF(MFs_Disponível)
11         Enviar_Dados_Migracao(MV_escolhida, MF_origem, MF_destino)
12     end
13 end
```

FIG. 4.4 Algoritmo Tratar Sobrecarga de Rede da MF

Ação Suportar Balanceamento de Carga - Esta ação é acionada pelo recebimento de mensagens como (MONITOR_INFO) ou (RELEVANT_EVENT) oriundos das MFs que compõem o cluster. As mensagens relativas a “RELEVANT_EVENT” são enviadas quando as máquinas não estão acometidas de nenhuma sobrecarga e ocorre uma diminuição ou aumento expressivo da utilização de recursos. A mensagem “MONITOR_INFO” é enviada sempre que o monitoramento de uma MF é iniciado ou quando ela recebe ou migra MVs. Estas contêm informações sobre os perfis de utilização dos recursos das MFs e MVs envolvidas e são analisados pelo Analisador.

De posse de uma destas mensagens, o algoritmo verifica o afastamento das CTMFs em relação à CMC, caso existam MF nesta situação, significa que a Carga Total de uma MF (CTMF) encontra-se fora da faixa dos valores permitidos para afastamento em relação à CMC. Constatada esta situação, o algoritmo escolhe a MF com maior valor de CTMF, esta será considerada a MF de origem. A partir desta MF, escolhe-se a MV de menor custo (MV escolhida) e cuja carga correspondente, quando retirada, permita que a MF (de origem) fique dentro ou próxima da Faixa de Equilíbrio. Em seguida, é escolhida uma

MF (de destino) com menor CTMF que possa receber a MV de modo que esta também fique dentro da Faixa de Equilíbrio ao receber a MV.

Depois de escolhida a MF de origem, a MV escolhida e a MF destino, é encaminhada a ordem de migração ao módulo Configurador.

Por consequência, este trabalho considera que o cluster está balanceado, quando todas as MFs do cluster encontram-se dentro da faixa de equilíbrio.

O afastamento em relação à CMC é configurado pelo administrador do cluster e deve permitir que a MF trabalhe com uma carga média próxima aos valores da CMC, mas sem gerar um exagerado número migrações de MVs. Ou seja, o administrador deve permitir um leve desbalanceamento do cluster, de modo a reduzir o número de migrações desnecessárias. Por exemplo, o administrador poderia considerar tolerável um afastamento de até 10% acima ou abaixo de um determinado valor de CMC. Diferenças maiores de 10% em relação ao valor CMC podem gerar migrações de MVs.

O afastamento descrito acima é definido, neste trabalho, como Faixa de Equilíbrio e pode ser entendida como a variação máxima da carga das MFs em relação à CMC. Quando uma MF fica fora dessa faixa, ela é considerada desequilibrada. Assim, devem ser tomadas medidas que envolvam a migração de MVs para que essa MF retorne a situação de equilíbrio.

A FIG. 4.5 apresenta o algoritmo que implementa o balanceamento de carga realizado pelo módulo Analisador.

```
Input: MONITOR_INFO,RELEVANT_EVENT, listaMonitores
Outpu ordemMigracao

// Realiza Balanceamento de Carga
receber_Mensagem_Monitor('MONITOR_INFO)
atualizarCTMF(MONITOR_INFO)
calculaCMC(lista_Monitores)
obterFaixadeEquilibrio(CMC)
listaAcimaCMC = verificarCTFMsAcima(lista_Monitores)
listaBaixoCMC = verificarCTFMsaBaixo(lista_Monitores)
Se  listAcimaCMC <>  vazio
    MFmaior = selecionaMFmaiorCTMF(lista_AcimaCMC)
    verificaMVdisponivel(MFmaior)
    Se listaMVdisponivel <> vazio
        MVmenor = selecionaMVmenorCTMF(MFmaior)
        Se MVmenor <> vazio
            migrarMV(MFmaio, MVmenor, MFmenorCTMF)
        fim se
    Senão
        msg='Não há MV disponivel para balanceamento'
Senão
    Se  listaBaixoCMC <>  vazio
        MFmaior = selecionaMFmaiorCTMF(lista_Monitores)
        listaMVDisponivel = verificaMVdisponivel(MFmaior)
        Se listaMVdisponivel <> vazio
            selecionaMFmenor CTMF(listaBaixoCMC)
            Se listaMFmenorValor <> vazio
                migrarMV(MFmaiorValor, Mvdisponviel, MFmenorCTMF)
            fim se
        Senão
            msg=' Não há MV disponível para balanceamento'
        Fim se
    Fim se
Fim se
```

FIG. 4.5 Algoritmo de Balanceamento de Carga

A FIG. 4.6 apresenta um exemplo do resultado do algoritmo de balanceamento de carga. Inicialmente, é apresentada a situação de cluster desbalanceado no tempo T1. Em seguida é mostrada a ocorrência de migrações de MVs no tempo (T2) e o balanceamento de cluster no tempo T3. Deste modo, todas as MFs ficam com suas CTMFs dentro da Faixa de Equilíbrio.

A Faixa de Equilíbrio é representada na FIG. 4.6, por 3 linhas horizontais. A linha laranja representa a carga média do cluster (CMC), que é o ponto

central da faixa, a linha vermelha abaixo da linha laranja representa o limite inferior da faixa e a linha vermelha acima da laranja representa o limite superior.

.

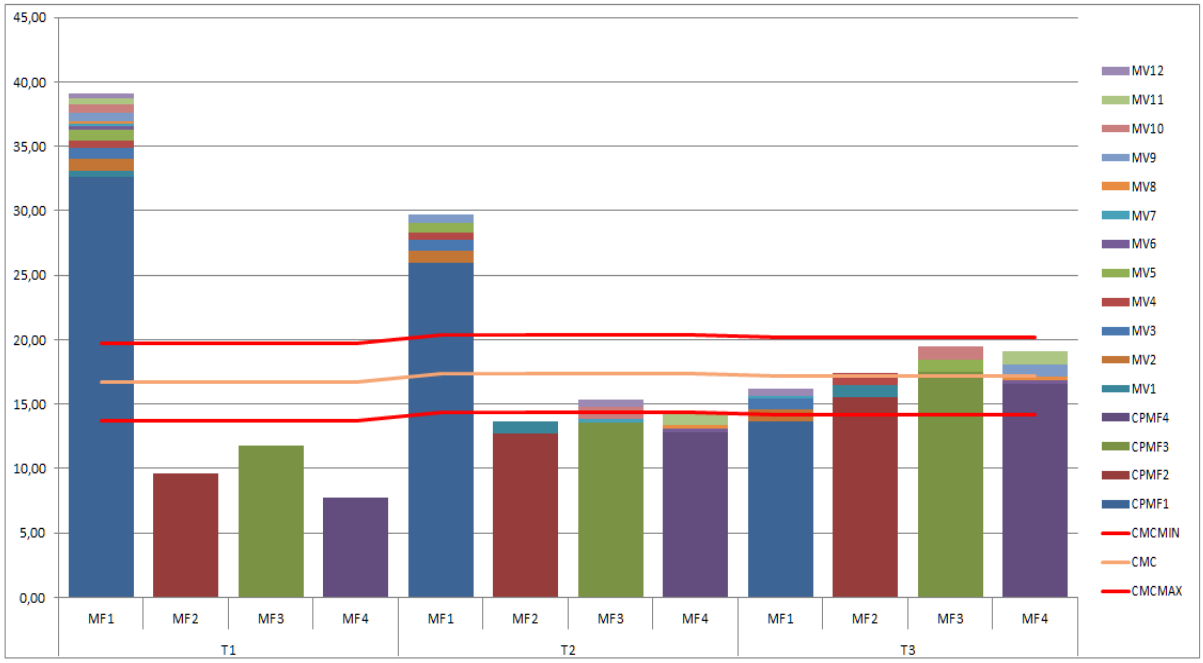

FIG. 4.6 Faixa de Equilíbrio da Carga das MF do Cluster

Ação Tratar Atingimento de ANS - O módulo Analisador estabelece que é necessário Tratar Atingimento de ANS pelo recebimento de mensagens (OVERLOADVM) oriundas de MFs que compõem o cluster. Estas mensagens são enviadas quando o percentual de uso da CPU, memória ou rede de uma MV chega e se mantém ao limite estabelecido pelo ANS. Neste caso, considera-se que esgotaram a capacidade de recursos (CPU, memória ou rede) contratados. O limite estabelecido para utilização dos recursos previstos em ANS é de 100%. Portando não se trata de uma restrição da MF hospedeira do serviço e sim do esgotamento dos recursos contratados.

Uma mensagem deverá ser enviada ao administrator para que ele possa tomar providências cabíveis junto ao contratante do serviço.

Para concluir as ações resultandes das mensagens enviadas pelo módulo Monitor, com exceção desta última, o módulo Analisador determina ao módulo Configurador que realize as migrações necessárias para a manutenção do cluster dentro das condições de funcionamento previstas.

4.2.2.3 MÓDULO CONFIGURADOR

Este módulo corresponde à fase de ação do ciclo OODA e transforma a estratégia de migração em comandos executáveis pelos hipervisores ou MMVs e determina sua execução por meio da API de virtualização.

Com base na estratégia estabelecida pelo módulo Analisador, o módulo Configurador codifica as instruções de modo que possam ser entendidas pelos MMVs envolvidos e determina sua execução. Este módulo utiliza a biblioteca libvirt para e execução das migrações necessárias por meio de comandos específicos.

Este módulo é responsável ainda, por parar o monitoramento antes das migrações das MVs, ativar o monitoramento depois da migração e informar ao módulo ao Analisador a finalização da migração determinada.

A FIG. 4.7 descreve as principais funcionalidades do algoritmo do módulo Configurador.

```
   Input:   Ordem de migração/dados_migração
   Output:  Situação Migração/situação atual da MF

 1 // Realiza migração   MV da MF origem e uma  de destino
 2 receber _dados_Migracao(MF_origem, MF_destino, MV_origem)
 3 paralizar_Monitor(MF_origem)
 4 realizar_Migração (MF_origem, MF_destino, Mv_origem)
 5 Se Migração == 'ok'
 6 |          envia_situacao_Migração("MIGRATION_FINISHED')
 7 senão
 8 |          envia_situacao_Migração('FALHA NA MIGRAÇÃO')
 9 end
10 inciar_Monitor(MF_origem)
```

FIG. 4.7 Algoritmo do Módulo Configurador

4.3 IMPLEMENTAÇÃO DA PHOENIX

Com o objetivo de validar o conjunto de procedimentos, descritos na Seção 4.2.2, foi configurado um ambiente computacional para a implementação da Phoenix. Esta Seção tem a finalidade apresentar as tecnologias utilizadas, para

suportar os algoritmos e os experimentos de avaliaram o comportamento da arquitetura nas diversas situações propostas.

A Phoenix foi implementada na linguagem de programação Python e utilizou a biblioteca libvirt (REDHAT, 2013) para se comunicar com as plataformas de virtualização e a biblioteca psutil (GOOGLE, 2013) para obter informações sobre a utilização dos recursos (CPU, memória, rede) das MFs e MVs.

A implementação foi feita baseada na programação com múltiplas *threads*, que monitoram e realizam o gerenciamento da infraestrutura virtualizada por meio de operações comando e controle.

Para cada MF, a Phoenix cria uma *thread* (Monitor) que monitora tanto a MF quanto as MVs hospedadas, colhendo informações de utilização de recursos (CPU, memória e rede). O intervalo de coleta é configurável e pode ser ajustado conforme a necessidade. Quando uma situação de sobrecarga ocorre ou um evento relevante é detectado, o Monitor envia uma mensagem à máquina central (Analisador).

Ao receber as mensagens dos vários Monitores, o módulo Analisador as avalia, observa os parâmetros estabelecidos no arquivo de configuração e determina a necessidade de movimentação ou fixação das MVs na MF original, conforme ações descritas na TAB 4.2. Estas ações são convertidas em orientações e enviadas ao módulo Configurador que efetivamente as implementa nas diversas plataformas por meio da libvirt (REDHAT, 2013). A principal vantagem desta API é permitir a comunicação com os principais hipervisores (MMVs) existentes.

4.4 ARQUIVO DE CONFIGURAÇÃO

O arquivo de configuração estabelece os parâmetros necessários ao funcionamento da arquitetura e pode ser descrito conforme a TAB. 4.5. Este arquivo é lido quando os módulos Monitor e Analisador entram em operação para obtenção de informações iniciais a respeito das MF, MV e configurações necessárias à parametrização destes módulos. Quando o cluster está em

operação, o Analisador é quem faz a leitura do arquivo para mudar parâmetros como, por exemplo, a Faixa de Equilíbrio do cluster.

TAB. 4.5 Parâmetros Necessários ao Funcionamento da Arquitetura Phoenix

Parâmetro	**Descrição**	**Valores Aceitos**	**Exemplo**
tipodeMaquina	Define o tipo da máquina que cuja a informação será avaliada	VIRTUAL/ FISICO	“VIRTUAL”
NAME	Nome da máquina	Alfanumérico	NAME=”MV1”
IP	IP da máquina	IP válido	IP=192.168.7.101
ansCPU	Quantidade de VCPUs da MV prevista no ANS	> 0	ansCPU=1
ansMEM	Quantidade de memória RAM da MV prevista no ANS	> 0	ansMEM=1024
ansREDE	Velocidade de transmissão de informações pela rede em Mbits/seg	> 0	ansREDE=250
limCPU	Percentual de utilização do recurso CPU previsto no ANS	1 -100	limCPU=100
limMEM	Percentual de utilização do recurso MEM previsto no ANS	1 -100	limMEM=100
limREDE	Percentual de utilização do recurso REDE previsto no ANS	1 -100	limREDE=100
variacaoCMC	Percentual aceitável de afastamento da CMC	0 -100%	variacaoCMC=8
ALFA	Constante utilizada na fórmula do custo da MV	0 -1	ALFA=0.4
BETA	Constante utilizada na fórmula do custo da MV	0 -1	BETA=0.6
capCPU	Quantidade de processadores (núcleos) da MF	> 0	capCPU=4
capMEM	Quantidade de Memória da MF medida em Megabytes	> 0	capMEM=4096
fixaMV	Indica se a arquitetura irá fixa a MV afetada pela sobrecarga da rede	SIM/NÃO	fixaMV = NAO
ordemMigracaoMV	Indica a ordem na qual a arquitetura irá migrar as MVs	maiorCusto/ menorCusto	ordemMigracaoMV= menorCusto
capREDE	Capacidade de Rede da MF medida em Mbits/seg	> 0	capREDE=1000

4.5 TROCA DE MENSAGENS ENTRE OS MONITORES E O ANALISADOR

A troca de mensagens ocorre entre os Monitores e o Analisador. Toda a troca de mensagens é feita por meio do protocolo UDP (*User Datagram Protocol*), conforme o descrito em Ferreira e Martins (2013).

O início se dá a partir do funcionamento do Analisador. Este, uma vez iniciado, fica em condições de receber mensagens dos Monitores. A cada evento relevante, os Monitores encaminham suas mensagens e de acordo com estas informações são feitos cálculos como: CTMF, CMC e definição da Faixa de Equilíbrio do Cluster. O resultado destes cálculos é atualizado na estrutura de dados do módulo Analisador, sendo esta estrutura utilizada para subsidiar a tomada de decisões deste módulo.

Os Monitores, por sua vez, ao iniciarem, enviam uma mensagem contendo os dados da MF e das MVs hospedadas (MONITOR_INFO). Com base nestas informações, o Analisador atualiza sua estrutura de dados e faz as tratativas, conforme descrito em 4.2.2.2.

Além das informações iniciais a respeito de sua configuração, os Monitores colhem informações de uso dos seus recursos e das MVs a cada 2 segundos (tempo configurável) e verificam a ocorrência de algum evento relevante. Caso ocorra um dos eventos descritos em 4.2.2.2, este envia uma mensagem ao Analisador informando o evento, as informações de uso de recursos da MF e de todas as MVs hospedadas. Deste modo, o Analisador se mantém constantemente atualizado.

As mensagens enviadas do módulo Monitor para o módulo Analisador podem ser dos seguintes tipos:

- RELEVANT_EVENT: indica a ocorrência de um Evento Relevante, conforme o descrito em 4.2.2.
- OVERLOADVM: indica que a MV atingiu o limite de utilização de recursos (CPU, memória ou rede) previsto no ANS e permanece no limite.
- OVERLOADPM: indica que um ou mais recursos atingiram o limiar de sobrecarga da MF.

- OVERNETVM: indica que o recurso rede de MVs ultrapassou o limiar configurado.
- MONITOR_INFO: indica o início das operações no Monitor ou reinício após uma migração. Contém informação sobre utilização dos recursos CPU, memoria e rede da MF e MV hospedadas.

O Analisador envia um tipo de mensagem para o módulo Configurador:

- MIGRATE: determina a migração de MV entre MFs

O Configurador envia a seguinte mensagem ao módulo Analisador:

- MIGRATION_FINISHED – Informa o final da migração da MV.

A FIG. 4.8, ilustra a troca de mensagens entre as máquinas ao longo da execução do algoritmo. O primeiro quadro indica envio da mensagem (MONITOR_INFO) dos Monitores para o Analisador, indicando o início de suas atividades. No segundo quadro, um dos monitores informa a ocorrência de um dos eventos relevantes ao Analisador (RELEVANT_EVENT). No quadro 3, o Analisador identifica que há necessidade de migração de MV e determina a migração de uma MV da MF de origem para uma MF de destino (MIGRATE). No quadro 4, a MF de origem inicia a migração. No quadro 5, a MF de destino informa o fim da migração (MIGRATION_FINISHED) e a MF de destino informa sua utilização de recursos e a utilização de recursos das MVs hospedadas (MONITOR_INFO).

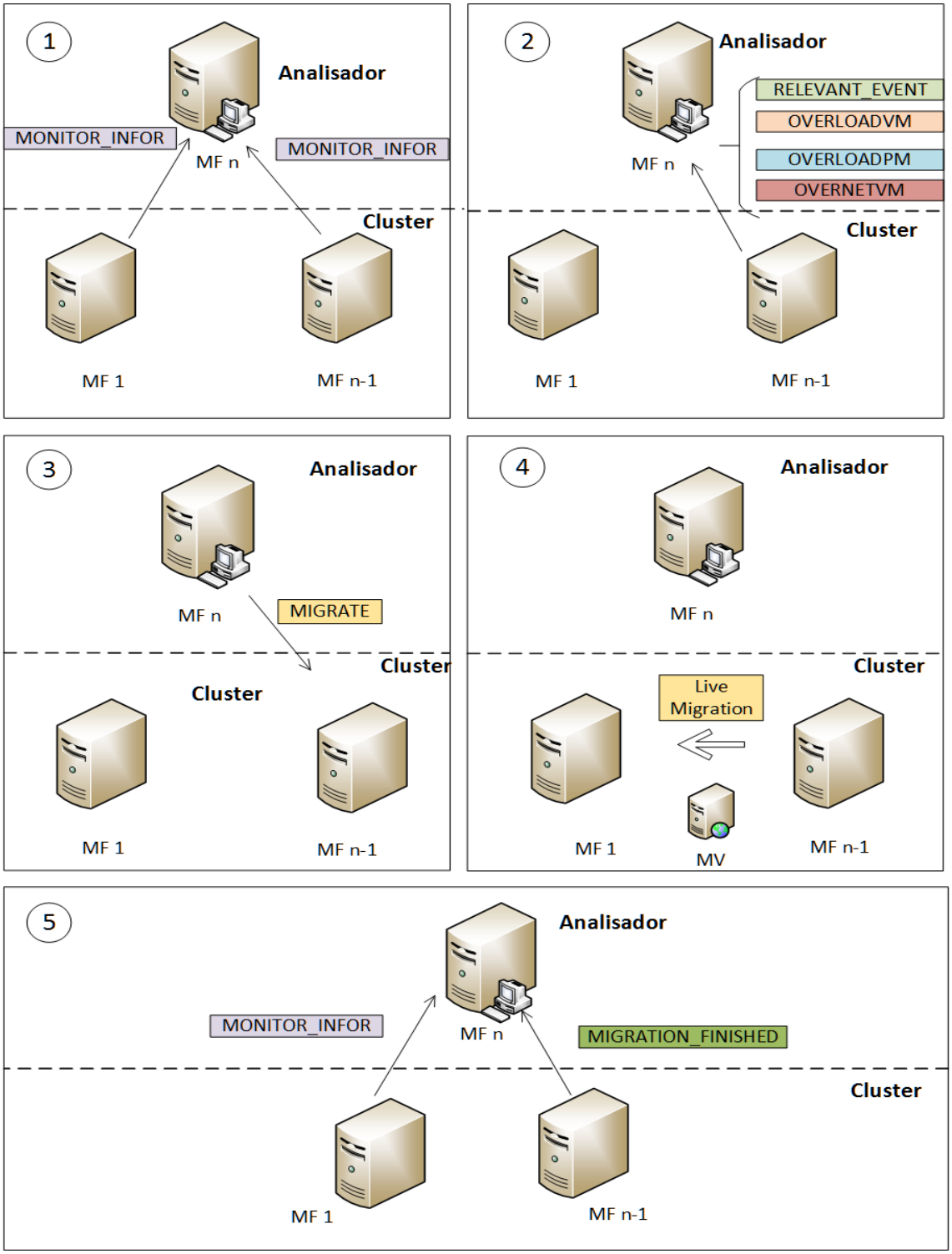

FIG. 4.8 Sequência de envio de Mensagens na Arquitetura Phoenix

5 EXPERIMENTOS REALIZADOS

Este capítulo tem a finalidade avaliar o comportamento da arquitetura Phoenix diante de sobrecargas momentâneas. Para tanto, foram selecionados três conjuntos de experimentos que demonstram a efetividade da proposta.

Inicialmente, é apresentada uma visão geral dos experimentos. Em seguida são descritos os experimentos e analisados os resultados de cada conjunto. Por fim, é apresentada uma análise geral dos resultados obtidos pela arquitetura proposta.

5.1 VISÃO GERAL

Com o objetivo de avaliar o comportamento da arquitetura Phoenix diante de sobrecargas momentâneas, foi proposto um conjunto de experimentos os quais foram realizados em um cenário restrito e real. Neste cenário, foram capturadas informações de perfis de uso reais, tanto das MFs quanto das MVs às quais foram submetidas ao módulo Analisador, proposto pela arquitetura, que indicou as tratativas necessárias para cada situação apresentada.

Os experimentos são descritos conforme se segue:

- Experimento 1: o primeiro conjunto de experimentos visou analisar o comportamento proativo da arquitetura, ou seja, balanceamento de carga. Isto permitiu identificar a Faixa de Equilíbrio adequada de funcionamento do cluster; e a parametrização escolhida se tornou referência para realização dos demais experimentos propostos.

- Experimento 2: o segundo conjunto de experimentos analisou o comportamento reativo da arquitetura quando submetida a sobrecarga dos recursos relacionados a CPU e a memória da MF.

- Experimento 3: o terceiro conjunto de experimentos avaliou o comportamento reativo da arquitetura quando submetida a sobrecarga do recurso rede da MV.

Os resultados foram analisados de forma qualitativa, sendo realizada a análise do comportamento da ferramenta em cada experimento.

5.2 AMBIENTE COMPUTACIONAL

Para a realização dos experimentos foi configurado um ambiente computacional que utilizou quatro servidores hospedeiros (MFs) idênticos (LabC202, LabC203, LabC204, LabC205), uma MF (LabC201) para execução do módulo Analisador e fornecimento do serviço de NFS (*Network File System*) [1]e uma MF (LabC206) utilizada como gerador de carga para simular as requisições dos clientes. Este ambiente pode ser observado na representação esquemática apresentada na FIG. 5.1.

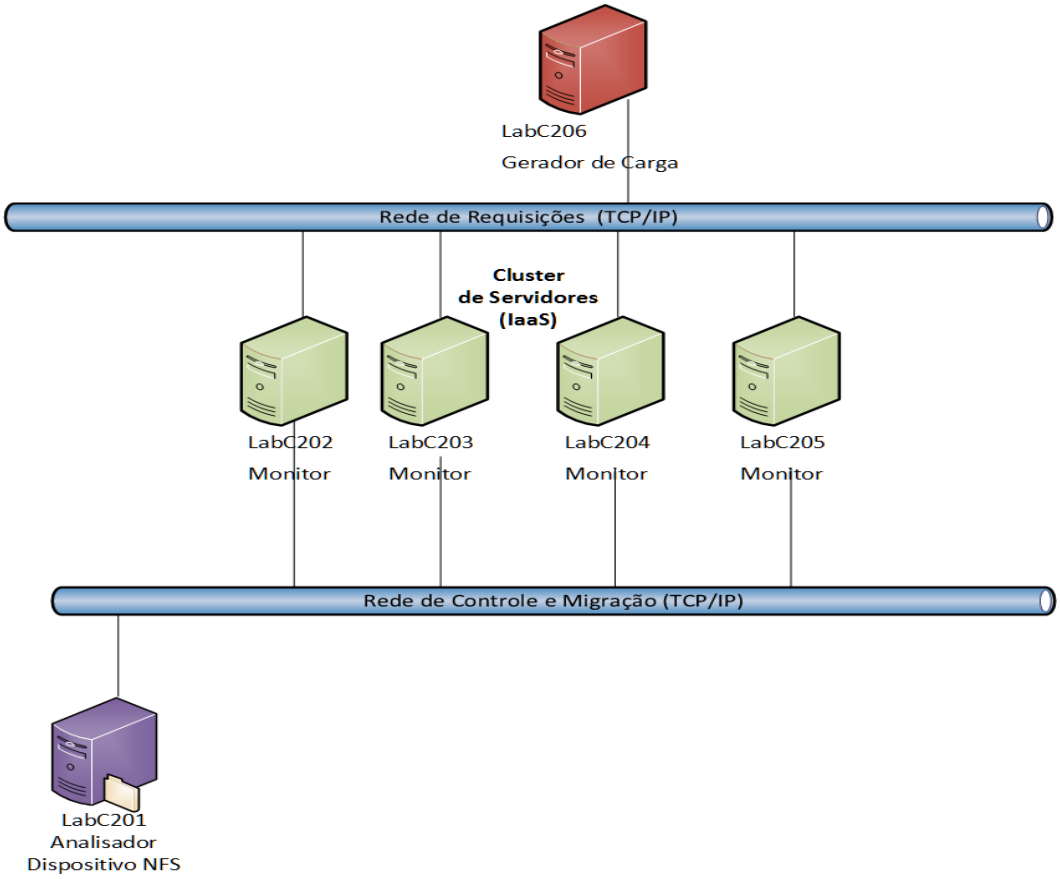

FIG. 5.1 Ambiente para a Execução dos Experimentos

Neste ambiente, foram estabelecidas duas redes gigabit ethernet: uma para requisições dos clientes (carga do simulador) e atendimento das requisições (Rede de Requisições) e outra para os serviços do próprio cluster

[1] Sistema de arquivo distribuído usado para compartilhar arquivos e diretórios entre sistemas interconectados, criando assim um diretório virtual

(Rede de Controle e Migração), tais como: serviço de NFS, controle e migração de MVs.

Os servidores hospedeiros possuem duas placas de rede para provimentos de comunicação em ambas as redes. O gerador de carga está ligado à Rede de Requisições de Clientes e o Analisador/Servidor NFS está ligado à Rede de Controle e Migração.

A descrição das MFs, MVs e Softwares utilizados na configuração do ambiente utilizado para os experimentos da arquitetura proposta são apresentados na TAB. 5.1.

TAB. 5.1 Descrição de Hardware e Softwares do Ambiente dos Experimentos

Componente da Arquitetura	**Softwares Utilizados**	**Parâmetros Utilizados**
Analisador (LabC201): Processador Intel®Core i5 E7500; 3 MB de Cache L2; 2.93 GHz; 1066 MHz FSB, 4 núcleos, RAM de 4 GB; Placa de rede de 1 Gbps	- Ubuntu 13.10 - Python 2.7 - SSH - NFS Server	IP=192.168.7.1
Gerador de Carga (LabC206): Processador Intel® Core2Duo com 3 MB de Cache L2; 3.0 GHz; 1066 MHz FSB, Ram de 4GB Placa de rede 1 – 1 Gbps	- Ubuntu 13.10 - Iperf (cliente)	IP=192.168.7.6
MF (LabC202 a LabC205): Processador Intel®Core i5 E7500; 3 MB de Cache L2; 2.93 GHz; 1066 MHz FSB, 4 núcleos, tecnologia VT (Vitualization Technology) habilitada RAM de 4 GB; Placa de rede – 1 Gbps Placa de rede 1 – 1 Gbps	- Ubuntu 13.10 - Python 2.7 - Libvirt 1.1.1 - Psutil 0.6.1-2 - Qemu-kvm.5.4 - NFS client - SSH - Stress 1.0.1 - Iperf (servidor)	IP=200.20.188.68 a 200.20.188.71 capCPU=4 capMEM=4096 capREDE=1000 IP=192.168.7.2 a 192.168.7.5 limCPU=80 limMEM=80 limREDE=80
MV (MV1 a MV12): VCPU - 1 Núcleo RAM - 1 GB Placa de rede – 250 Mbp/s	- Ubuntu 13.10 - Stress 1.0.1 - Iperf (servidor)	NAME=”MV1 a “MV12” IP=192.168.7.101 a 192.168.7.112 ansCPU=1 ansMEM=1024 ansREDE=250

Nos servidores hospedeiros (MFs), foram instaladas 12 (doze) MVs. As MVs estão hospedadas nas MFs aqui representadas pelas LabC202 até LabC205.

As MVs foram configuradas com as quantidades máximas de recursos de processamento (referenciados em unidades de processamento virtual (VCPU)), de rede (referenciado em megabits por segundo), e de memória (referenciado em megabytes), que correspondem às capacidades máximas estabelecidas nos ANS de cada infraestrutura como serviço (IaaS) hospedada.

Os limites para utilização dos recursos (CPU, memória e rede) das MVs correspondem a 100% da capacidade acordada nos respectivos ANSs.

No caso das MFs, o limite de utilização estabelecido é de 80% para os recursos CPU e rede, e 70% para o recurso memória. São considerados nestes limites os recursos utilizados tanto pelas MVs quanto pelas próprias MFs hospedeiras. Percentuais de utilização de recursos acima dos limiares estabelecidos são considerados como sobrecarga.

De modo geral, os limiares estabelecidos visam garantir que haja recursos para o funcionamento normal do sistema operacional das MFs, da plataforma de virtualização, dos serviços hospedados e dos módulos componentes da arquitetura. No caso da memória, o limiar foi estabelecido em 70% porque este limiar se mostrou mais eficaz em razão dos testes realizados. Valores acima deste podem comprometer a migração da MV em razão da excessiva utilização da memória em disco (*swap*), quando a MF sofre uma sobrecarga do recurso memória.

Inicialmente, a arquitetura foi calibrada com informações, tais como: capacidades de recursos totais previstos para cada MF, capacidades de recursos das MVs previstas nos ANS, limiar de sobrecarga de recursos (CPU, utilizados pelas MFs que foram utilizados nos cálculos da Carga Total da MF (CTMF) e carga média do cluster (CMC), conforme o descrito na TAB. 4.5.

Para gerar carga na CPU e memória das MFs e MVs, foi utilizado o aplicativo Stress (AMOS WATERLAND, 2013) enquanto que, para gerar carga na rede das MVs, foi utilizado o aplicativo iperf (IPERF, 2014).

5.3 EXPERIMENTO 1 – IDENTIFICAÇÃO DA FAIXA DE EQUILÍBRIO

O balanceamento de carga, no contexto deste trabalho, pode ser entendido como uma ação proativa em relação à sobrecarga da MF, pois evita esgotamento prematuro dos recursos computacionais por meio da distribuição de cargas entre as MFs do cluster.

Quando o cluster está em situação normal, seu acionamento se dá por meio da identificação de MFs com cargas totais (CTMF) com variação acima ou abaixo do limite configurado para o cluster, ou seja, fora da Faixa de Equilíbrio.

Porém, o excesso de migrações de MVs pode provocar aumento da utilização de recursos das MFs e sobrecarga da rede. Por isso, há necessidade de se obter uma variação de CMC que permita um número de migrações reduzido, mas que efetivamente deixe o cluster balanceado.

Este experimento teve por finalidade estabelecer a parametrização da Faixa de Equilíbrio de modo que as MFs fiquem balanceadas sem que seja necessário um número excessivo de migrações.

Para definição da Faixa de Equilíbrio do Cluster, o algoritmo foi parametrizado com quatro opções de percentuais de afastamento em relação à CMC. As MVs do cluster foram distribuídas de modo a tornar o cluster desequilibrado. Ao detectar a necessidade e executar o balanceamento de carga das MFs, o algoritmo proposto movimentou MVs de modo a balancear o cluster.

Para escolha da Faixa de Equilíbrio adequada, foi contabilizado a quantidade de migrações efetuadas, o tempo gasto e a uniformidade da distribuição das MVs, sendo escolhida a variação que apresentou o menor tempo total, o menor número de migrações e obteve uma distribuição adequada das MVs.

Para realizar este experimento foram criadas doze MVs em uma única MF (Labc202). Como as MVs hospedadas não estavam rodando aplicações que utilizavam grande parte de seus recursos previstos nos ANSs, foi possível hospedá-las em uma única MF sem causar sobrecarga no servidor (Labc202). Somente a memória da MF ficou próxima do limite de sobrecarga (80%), conforme o apresentado na FIG. 5.2.

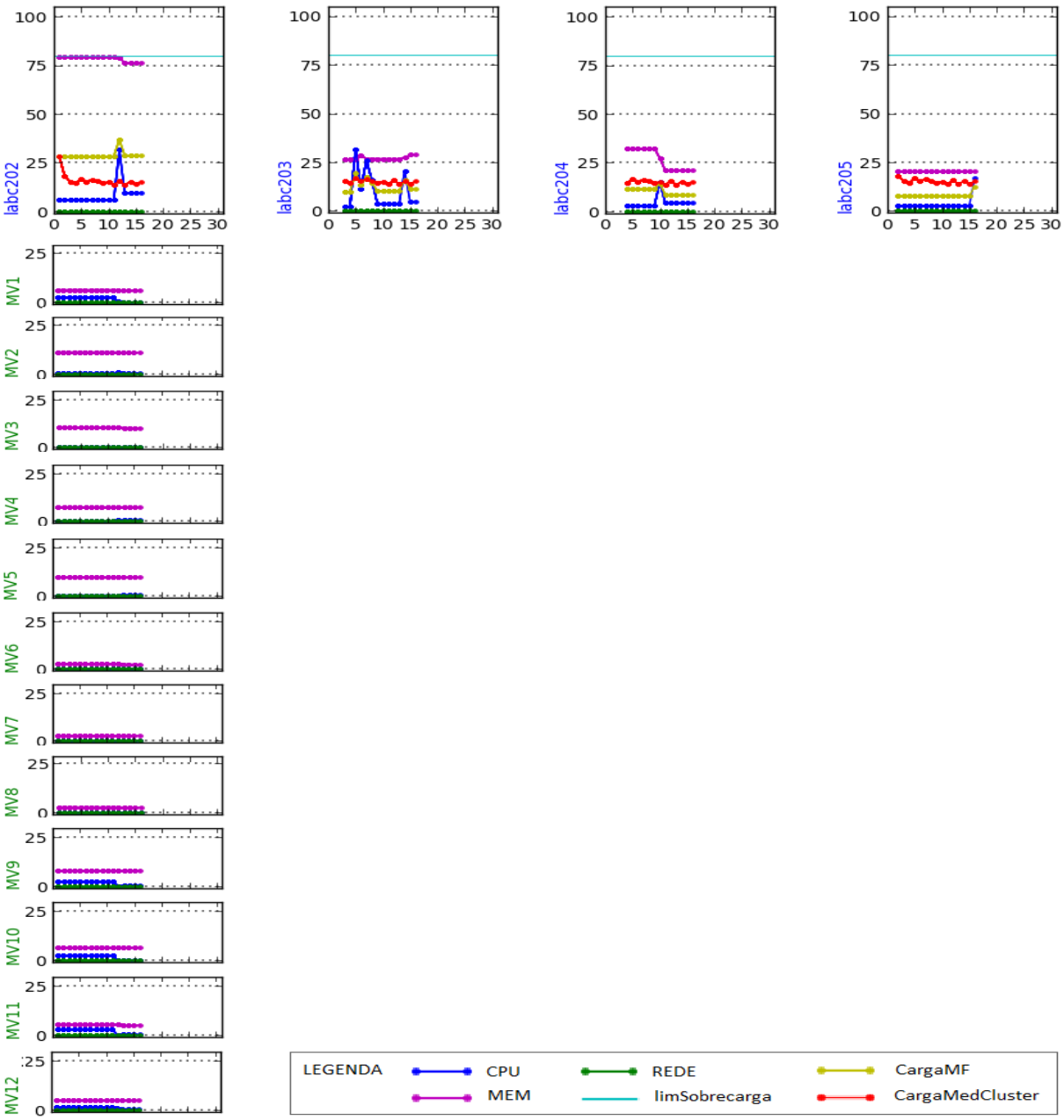

FIG. 5.2 Configuração inicial do cluster com todas as MVs em uma única MF

A diagramação apresentada na FIG. 5.2 foi utilizada para a representação do comportamento da arquitetura diante dos diversos experimentos realizados. Dessa forma, visando à compreensão dos resultados apresentados, os elementos são:

- Máquinas Físicas: na parte superior do diagrama, em linha, estão dispostas as MFs do cluster. Para cada MF, o gráfico apresenta os recursos utilizados (CPU, MEM, REDE). Ainda neste gráfico, são apresentados a CTMF, CMC e limite de Sobrecarga configurado. Os recursos são medidos ao longo do eixo horizontal, com escala de 0 a 30 pontos que representam as medidas realizadas como consequência de mensagens geradas por eventos relevantes, e do eixo vertical, cuja escala varia de 0 a 100 e representa o percentual de utilização dos recursos da MF. Assim, a configuração adotada permite que até 30 eventos relevantes sejam computados no gráfico.
- Máquinas Virtuais: abaixo de cada MF e alinhadas com ela, são apresentadas as MVs alocadas nesta MF. De modo semelhante ao que ocorre com as MFs, são apresentados no gráfico de cada MV os recursos utilizados, medidos ao longo de eventos relevantes. Ao ocorrer uma atualização de medição de qualquer MF, os recursos utilizados pelas MVs contidas nesta, também são atualizados. O eixo vertical representa percentual de uso de recursos utilizados pela MV, destacando as faixas de 25%, 50%, 75% e 100%, de acordo com a utilização da MV, visando simplificar o diagrama.
- Utilização de Recursos: a legenda apresenta traços com cores distintas representando o percentual de utilização dos recursos de CPU (azul), memória (magenta) e rede (verde), tanto das MVs quanto das MFs. Na legenda, o traço seguinte (ciano) representa o limite de sobrecarga das MFs, que está definido em 80% da capacidade total para CPU, memória e rede. Em seguida encontra-se o traço que indica a CTMF (amarelo), e por último, a CMC (vermelho). Por meio destes pontos, é possível perceber a evolução da utilização dos recursos (CPU, memória e rede) das MFS e MVs ao da ocorrência de

eventos relevantes. Também é possível acompanhar a evolução da CMC e CTMFs.

Como pode ser observado na FIG. 5.2, a MF Labc202, ao hospedar as doze MVs (MV1 a MV12), teve principalmente seu percentual de utilização da memória aumentado contribuindo para seu afastamento da CMC. Nesta condição, o Analisador verifica o afastamento desta MF da Faixa de Equilíbrio e procede, conforme o descrito na Seção 4.2.2.2, realizando o balanceando do cluster.

De modo resumido, caso a carga total da MF (CTMF) analisada esteja fora dos limites de tolerância (afastamento da CMC definido em pontos percentuais), o Analisador seleciona e determina migrações de MVs, de modo que ao final da operação todas as MF fiquem com suas cargas totais (CTMF) dentro do limite estabelecido (Faixa de Equilíbrio). Feito isso, o cluster encontra-se balanceado.

Os pontos percentuais utilizados na identificação da CMC são: (a) 3, (b) 5, (c) 8 e (d) 10 em relação à CMC. Os resultados são descritos como se segue.

5.3.1 VARIAÇÃO DE 3 PONTOS PERCENTUAIS EM RELAÇÃO À CMC

Para caracterizar o uso dos parâmetros, foram colocados em destaque na TAB. 5.2 os parâmetros extraídos da TAB. 4.5 e configurados para realização deste experimento.

TAB. 5.2 Parâmetros para variação igual a 3 pontos percentuais

Parâmetro	**Descrição**	**Valores Aceitos**	**Valor para o Experimento**
ordemMigraçãoMV	Indica a ordem na qual a arquitetura irá migrar a MVs	maiorCusto/ menorCusto	ordemMigracaoMV= menorCusto
variacaoCMC	Variação permitida em relação à CMC	1 a 100	variacaoCMC = 3 pontos percentuais

Com os parâmetros acima configurados, partindo da situação descrita na FIG. 5.2, identificada a necessidade de balanceamento pelo módulo Analisador, a arquitetura começa a fazer migrações, distribuindo MVs entre as MFs do cluster. A FIG. 5.3 retrata a distribuição de MVs em um momento intermediário, após 3 MVs terem sido migradas.

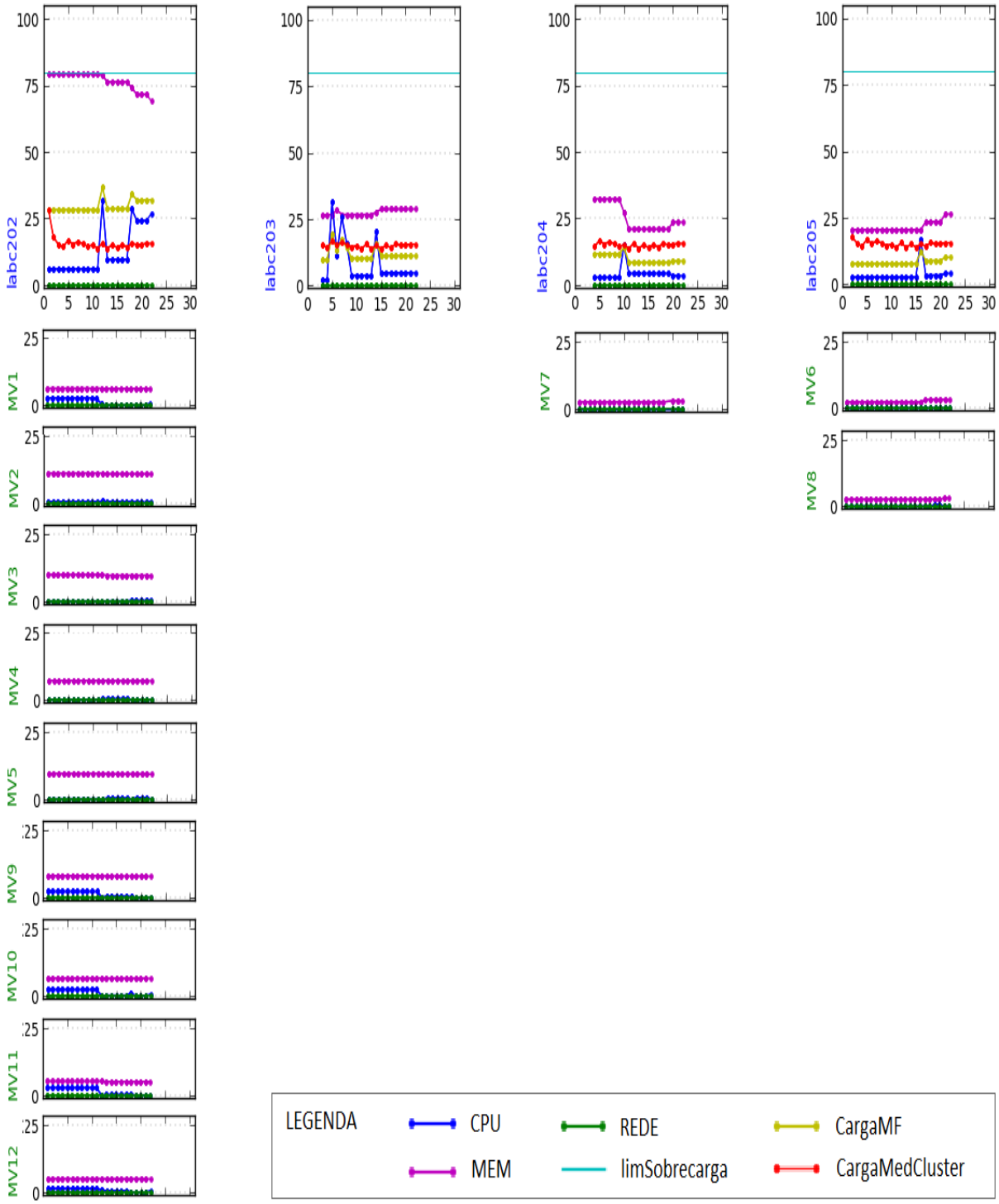

FIG. 5.3 Configuração após 3 migrações - variação de 3 pontos

Observa-se na FIG. 5.3, que, após migrar 3 MVs, a Lac202 apresentou queda na utilização do recurso memória contribuindo para aproximar sua CTMF da CMC. Em relação à MF Labc203, nota-se que a mesma estava com a sua

CTMF mais próxima da CMC, por isso não recebeu, até este momento, nenhuma MV. Quanto às demais MFs do cluster, nota-se que houve um ligeiro aumento da proximidade de suas CTMF com a CMC, em virtude das migrações de MVs.

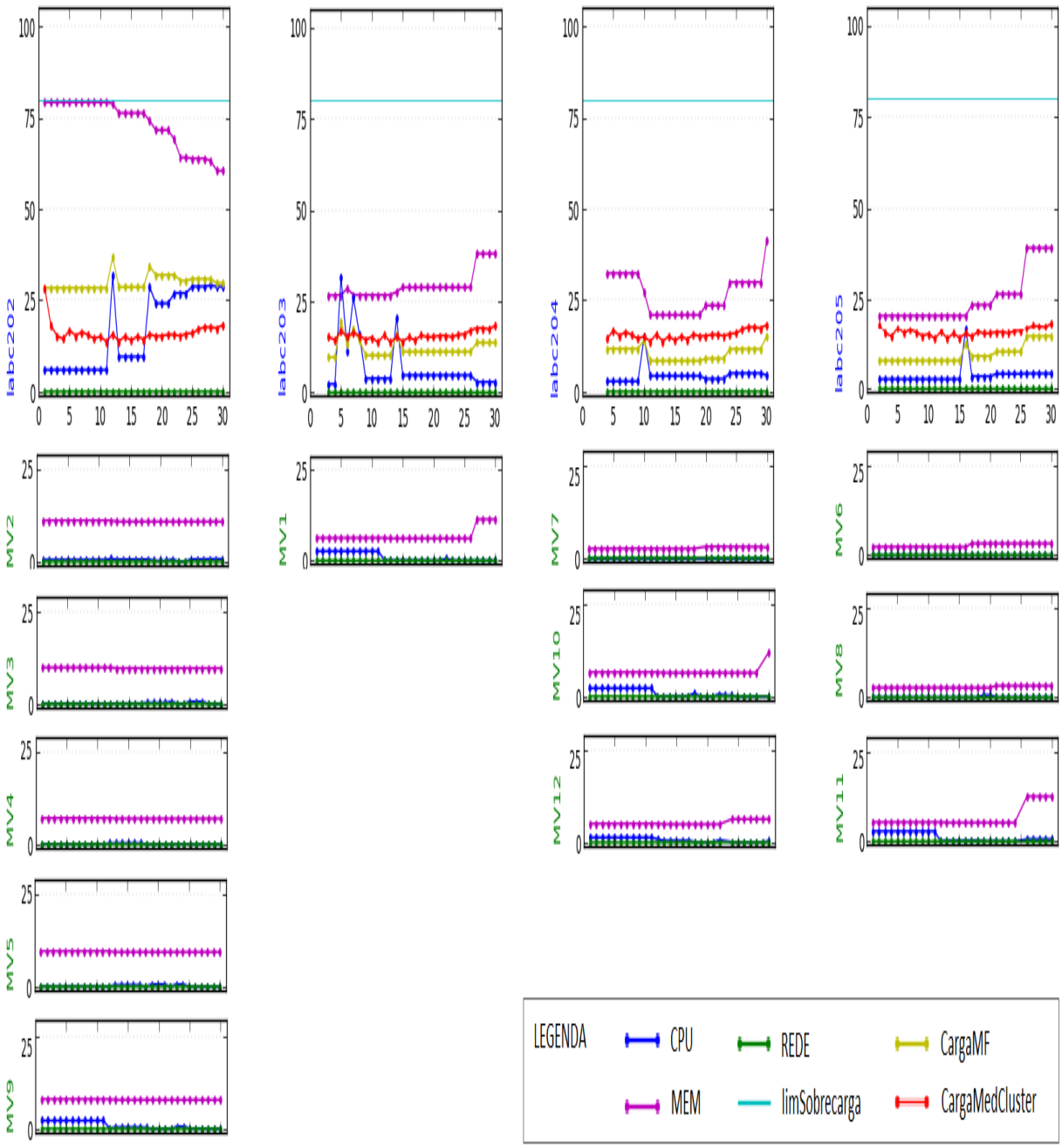

FIG. 5.4 Configuração após 7 migrações - variação de 3 pontos

Após 7 migrações de MVs, apresentadas na FIG. 5.4 observa-se, em relação à MF Labc202, uma queda acentuada do recurso memória e a

aproximação de sua CTMF com a CMC. Em relação às demais MFs do cluster, nota-se que a Labc203 manteve sua CTMF próxima da CMC, enquanto que as MFs Labc204 e Labc205 tiveram uma aproximação ainda maior de suas CTMFs da CMC motivado pela migração de MVs.

Observa-se na FIG. 5.5 que, após 13 migrações de MVs, realizadas em 3 minutos e 5 segundos, a arquitetura finalizou o balanceamento de carga. Nota-se em relação à Labc202 que, além da acentuada queda da utilização do recurso memória, os demais recursos (CPU e rede) se mantiveram em condições similares às demais MFs do cluster, havendo uma aproximação de suas CTMFs com a CMC. Em relação às demais MFs do cluster, nota-se que todas alcançaram uma proximidade ainda maior de suas CTMFs com a CMC mesmo com a adição de novas MVs por ocasião das migrações realizadas. Assim, como a variação de 3 pontos percentuais (em relação ao valor total da CMC) foi obedecida por todas as MFs, o cluster encontra-se balanceado.

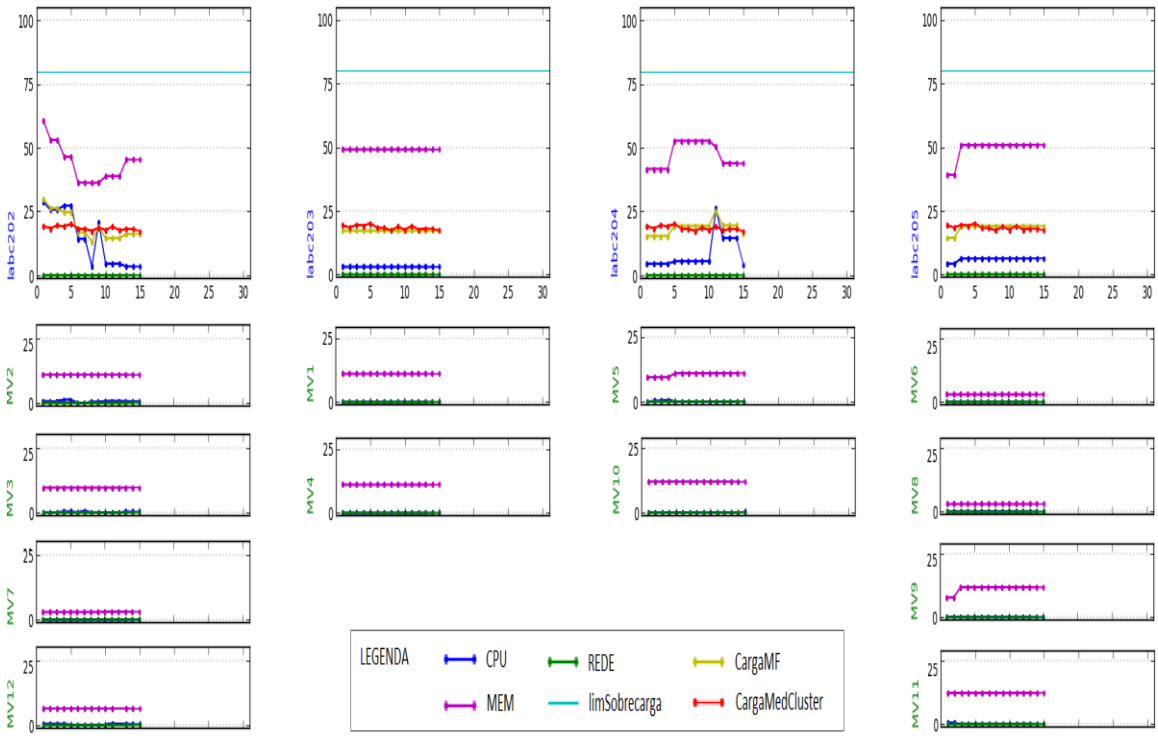

FIG. 5.5 Configuração Final utilizando variação de 3 pontos percentuais

5.3.2 VARIAÇÃO DE 5 PONTOS PERCENTUAIS EM RELAÇÃO À CMC

Para variação de 5 pontos percentuais, proposta neste experimento, foram colocados em destaque na TAB. 5.3 os parâmetros da TAB. 4.5 modificados para realização deste experimento:

TAB. 5.3 Parâmetros para variação igual a 5 pontos percentuais

Parâmetro	Descrição	Valores Aceitos	Exemplo
ordemMigraçãoMV	Indica a ordem na qual a arquitetura irá migrar as MVs	maiorCusto/menorCusto	ordemMigracaoMV= menorCusto
VariacaoCMC	Variação permitida em relação à CMC	1 a 100	variacaoCMC = 5

Com este percentual configurado, partindo de uma situação semelhante à descrita na FIG. 5.2, após 5 migrações de MVs, é apresentada a distribuição das MFs do cluster estabelecida em um momento intermediário .

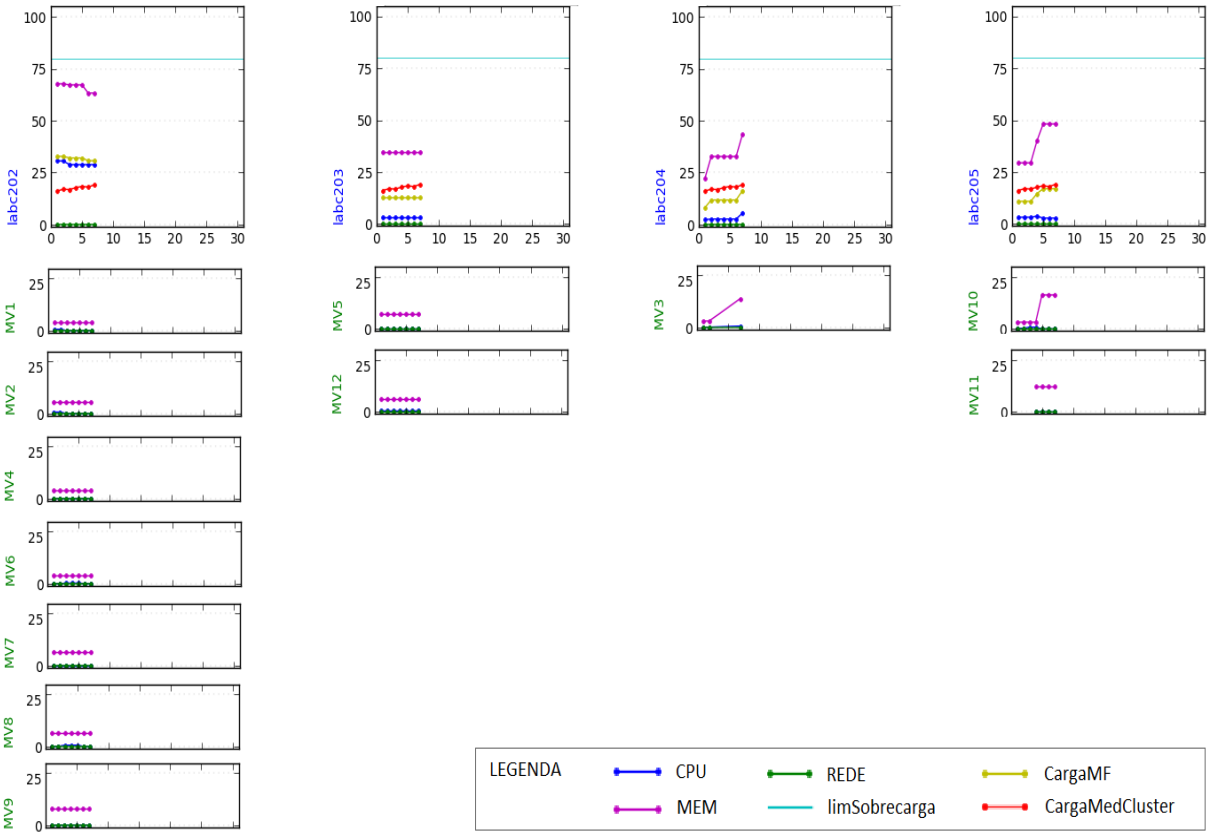

FIG. 5.6 Configuração após 5 migrações utilizando a variação 5

Conforme pode ser visto na FIG. 5.6, a MF Lac202, após migrar 5 MVs, apresentou queda na utilização do recurso memória, fato que contribuiu para aproximação da sua CTMF com a CMC. Em relação à MF Labc203, nota-se que mesmo recebendo 2 MVs, a MF manteve um pequeno distanciamento de sua CTMF em relação à CMC. Quanto às demais MFs do cluster (Labc204 e Labc205), nota-se que houve a proximiação de suas CTMF com a CMC.

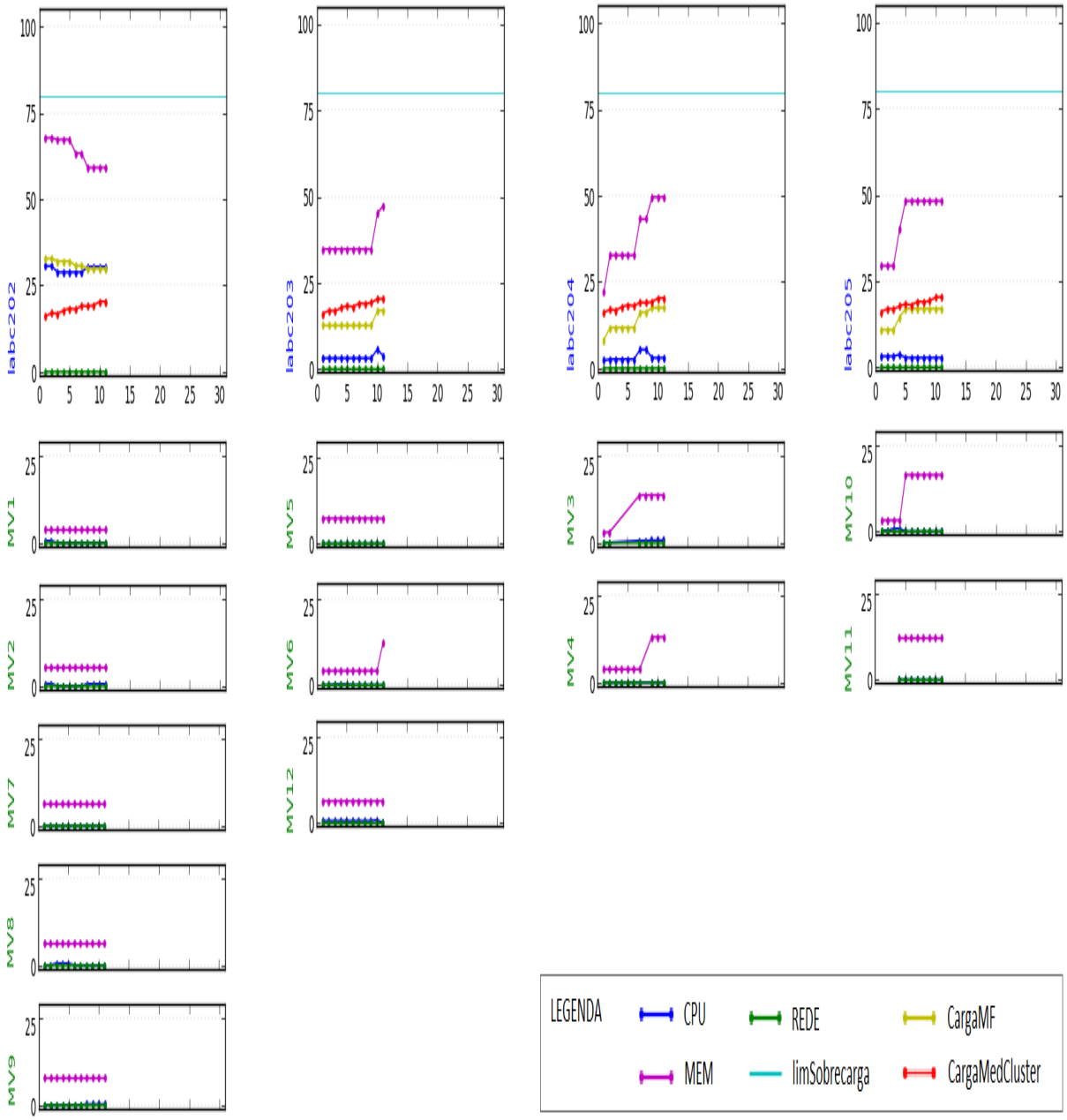

FIG. 5.7 Configuração após 7 migrações utilizando a variação 5

Após 7 migrações, conforme pode ser verificado na FIG. 5.7, a MF Lac202, ampliou a queda da utilização do recurso rede contribuindo para aproximação da sua CTMF com a CMC. Em relação à MF Labc203, nota-se que após receber a terceira MV, obteve uma aproximação de sua CTMF em relação à CMC. Quanto às demais MFs do cluster (Labc204 e Labc205), nota-se que houve a manutenção da proximidade de suas CTMF com a CMC.

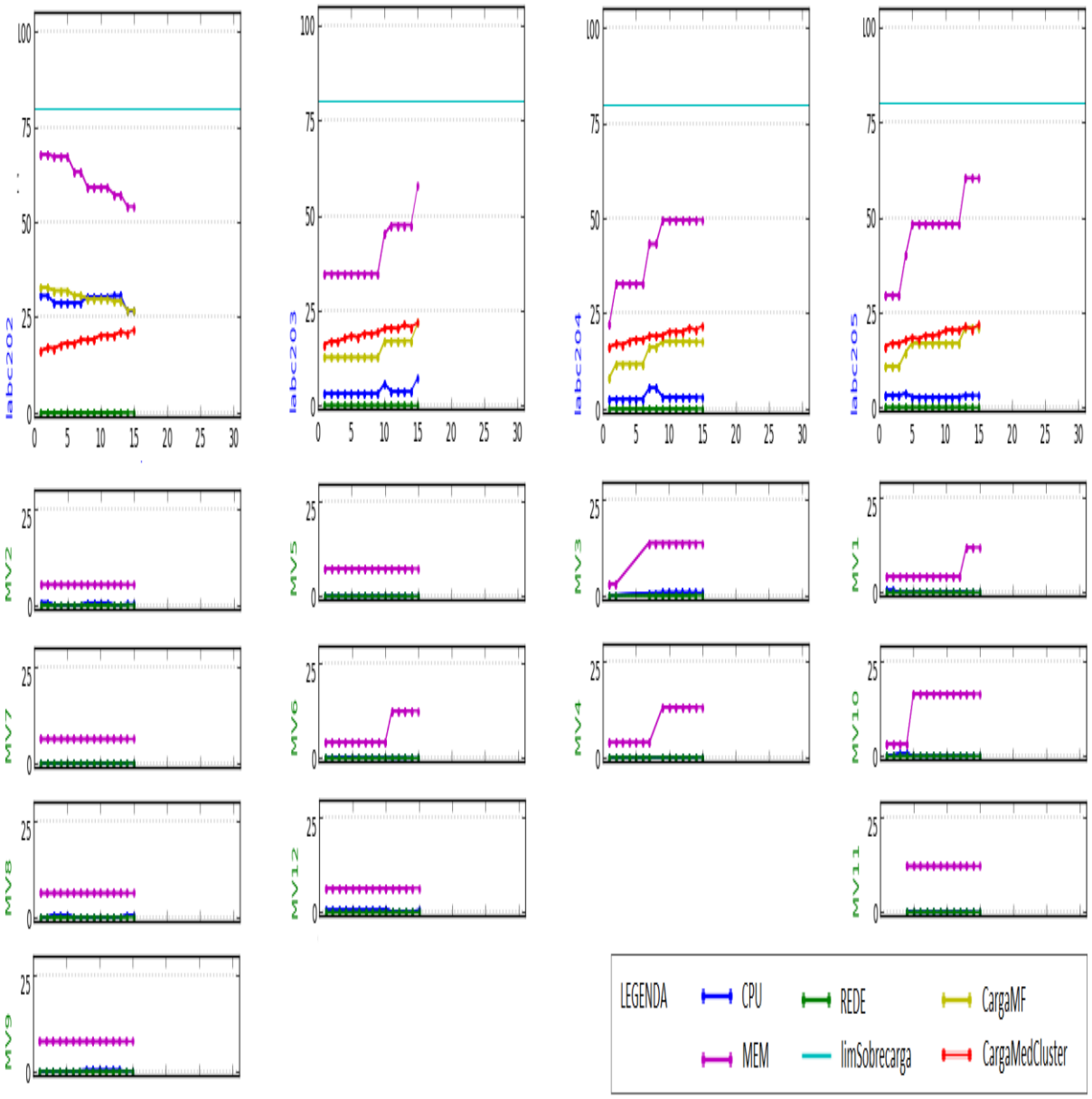

FIG. 5.8 Configuração Final do cluster utilizando da variação 5

Após 8 migrações de MVs, realizadas em 2 minutos e 38 segundos, o algoritmo finalizou o balanceamento de carga, conforme mostrado na FIG. 5.8. Observa-se em relação à Labc202 que, além da queda acentuada do recurso memória, os demais recursos (CPU e rede) se mantiverem em condições compatíveis com as demais MFs do cluster, havendo uma aproximação de sua CTMF com a CMC. Em relação às demais MFs do cluster, nota-se que todas alcançaram uma proximidade ainda maior de suas CTMF com a CMC, mesmo com a adição de novas MVs, por ocasião das migrações realizadas. Assim, o cluster encontra-se balanceado.

5.3.3 VARIAÇÃO DE 8 PONTOS PERCENTUAIS EM RELAÇÃO À CMC

Para variação de 8 pontos percentuais, proposta neste experimento, foram colocados em destaque na TAB 5.4 os parâmetros da TAB. 4.5 modificados para realização deste experimento.

TAB. 5.4 Parâmetros para Variação igual a 8 pontos percentuais

Parâmetro	**Descrição**	**Valores Aceitos**	**Exemplo**
ordemMigraçãoMV	Indica a ordem na qual a arquitetura irá migrar a MV	maiorCusto/ menorCusto	ordemMigracaoMV= menorCusto
VariacaoCMC	Variação permitida em relação à CMC	1 a 100	VariacaoCMC = 8

Com este percentual configurado, partindo de uma situação semelhante à descrita na FIG. 5.2, são verificadas as moviemntações das MVs entre as MFs participantes do cluster e descritas como se segue.

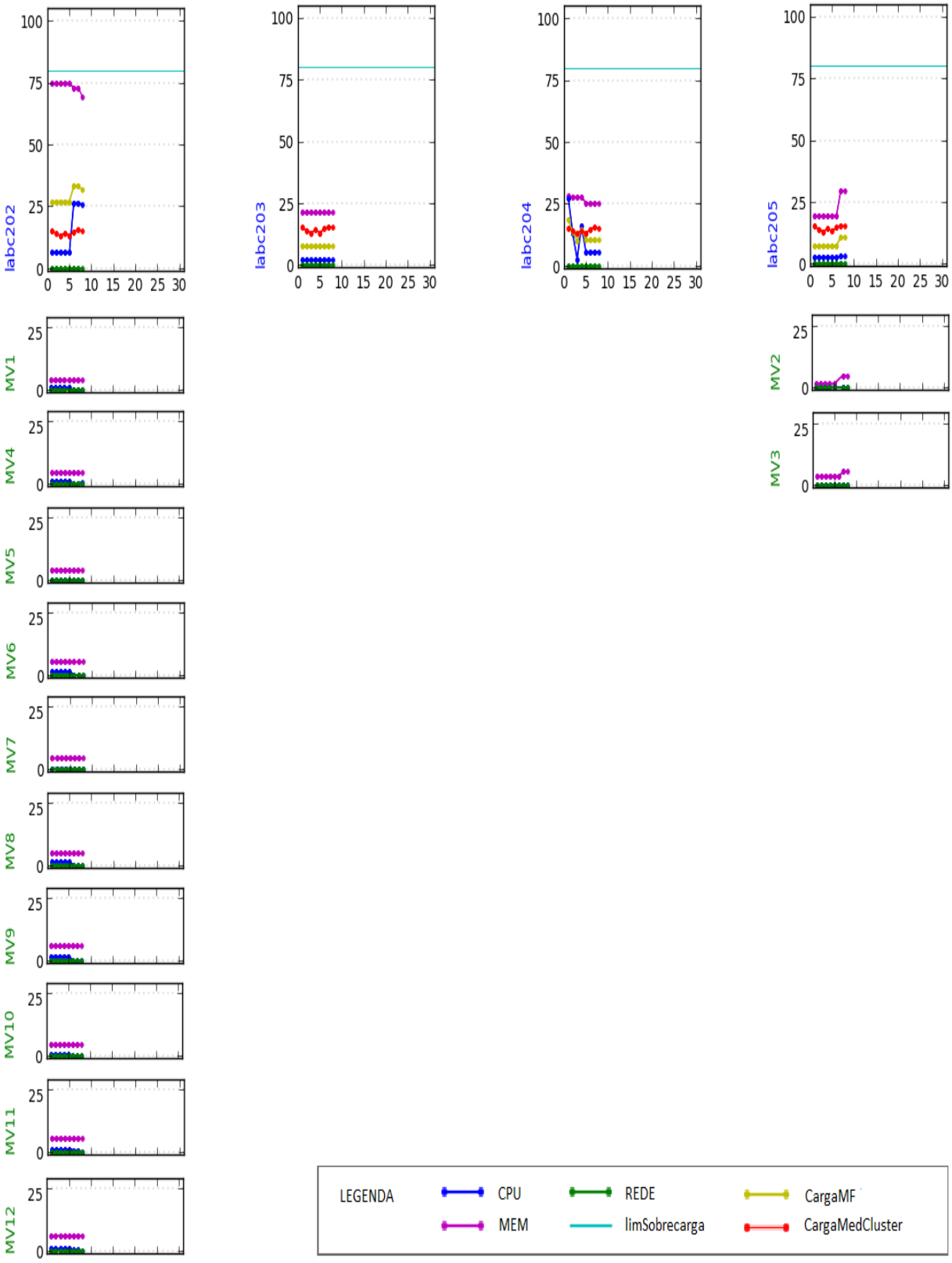

FIG. 5.9 Configuração Intermediária (1) utilizando a variação 8

Conforme pode ser visto na FIG. 5.9, a MF Lac202, após migrar 2 MVs, apresentou uma leve queda na utilização do recurso memória contribuindo para aproximação da sua CTMF com a CMC. Em relação às MFs Labc203 e Labc204, nota-se que estas mantiveram um pequeno distanciamento de suas CTMFs em relação à CMC. Quanto a Labc205, nota-se que, após a migração de 2 MVs, houve uma maior aproximação de sua CTMF com a CMC.

.

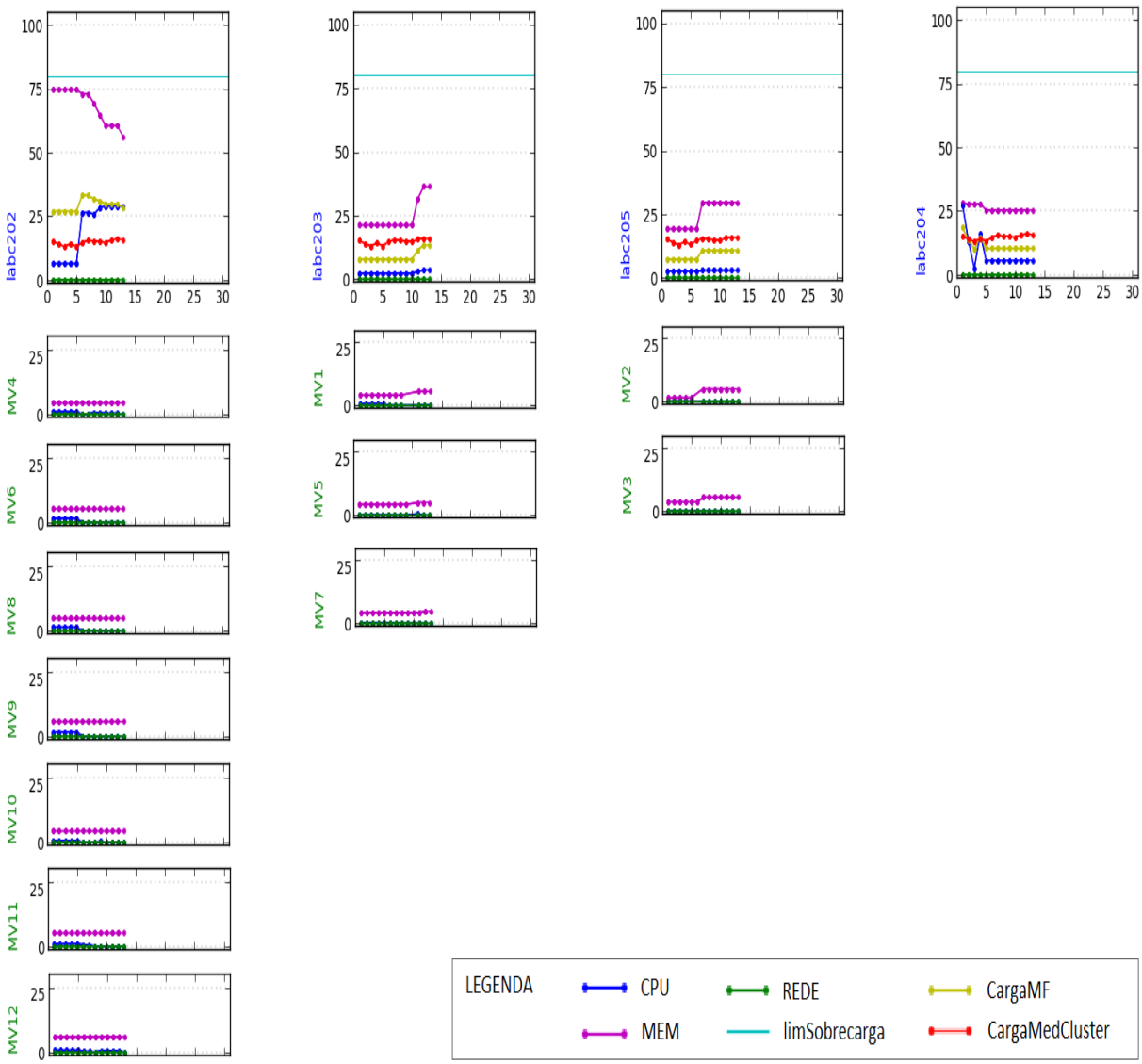

FIG. 5.10 Configuração Intermediária (2) utilizando a variação 8

Conforme pode ser visto na FIG. 5.10, a MF Lac202, após migrar 5 MVs, apresentou uma queda acentuada na utilização do recurso memória

contribuindo para aproximação da sua CTMF com a CMC. Em relação à MF Labc203 e Labc204, nota-se que após uma aproximação mantiveram um pequeno distanciamento de suas CTMFs em relação à CMC. Quanto a Labc205, nota-se que após a migração de 2 MVs, houve um pequeno afastamento de sua CTMF com a CMC.

Após 8 migrações de MV realizadas em 1 minuto e 19 segundos, a arquitetura finalizou o balanceamento de carga, conforme mostrado na FIG. 5.11. Observa-se em relação à Labc202 que, além da queda acentuada do recurso memória, os demais recursos (CPU e rede) se mantiverem em condições compatíveis com as demais MFs do cluster, houve uma aproximação muito expressiva da CTMF com a CMC, chegando ao ponto de se tocarem. Em relação às demais MFs do cluster, nota-se que todas alcançaram uma proximidade ainda maior de suas CTMF com a CMC, após as migrações realizadas. Dessa forma, o cluster encontra-se balanceado.

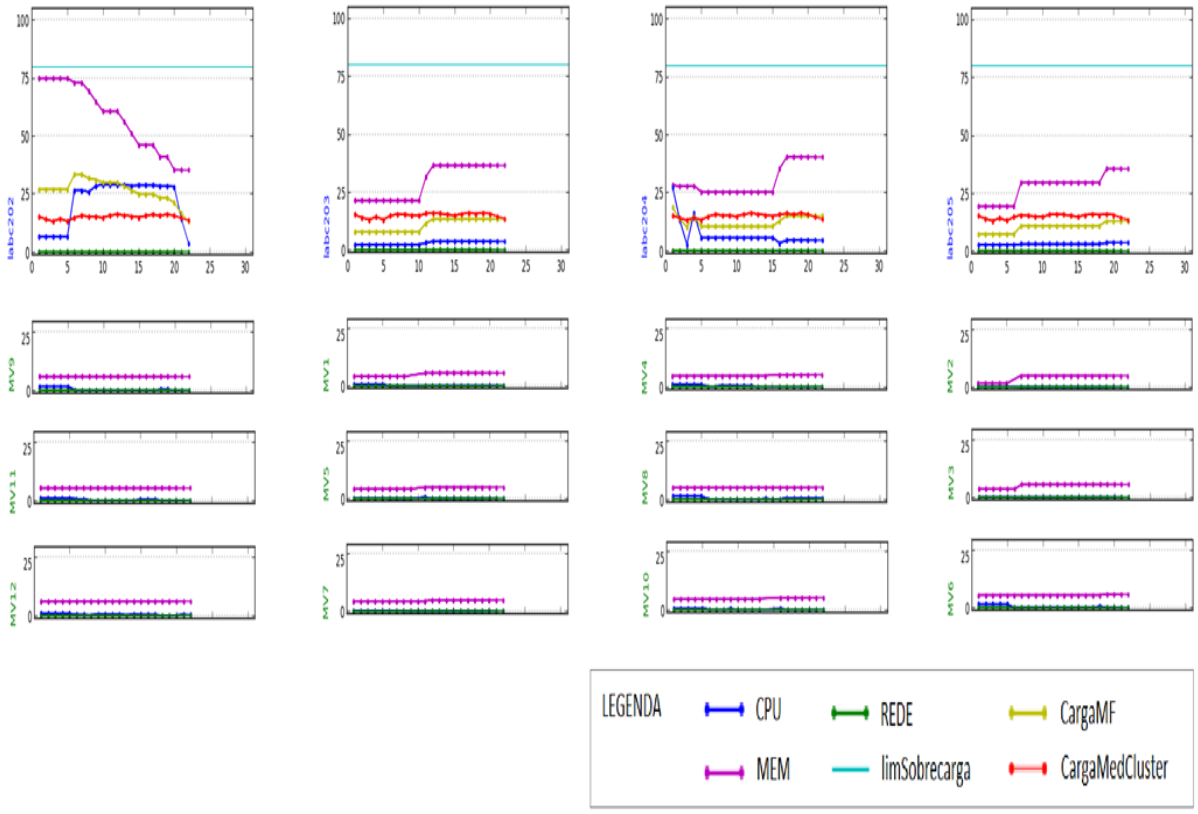

FIG. 5.11 Configuração Final do cluster utilizando a variação 8

5.3.4 VARIAÇÃO DE 10 PONTOS PERCENTUAIS EM RELAÇÃO À CMC

Para variação de 10 pontos percentuais, proposta neste experimento foram colocados em destaque na TAB. 5.5 os parâmetros da TAB. 4.5 modificados para realização deste experimento.

TAB. 5.5 Parâmetros para variação igual a 10 pontos percentuais

Parâmetro	**Descrição**	**Valores Aceitos**	**Exemplo**
ordemMigraçãoMV	Indica a ordem na qual a arquitetura irá migrar as MVs	maiorCusto/ menorCusto	ordemMigracaoMV= menorCusto
VariacaoCMC	Variação permitida em relação à CMC	1 a 100	variacaoCMC = 10

Com este percentual configurado, partindo de uma situação semelhante à descrita na FIG. 5.2, após 3 migrações de MVs, é apresentada na FIG. 5.12 a distribuição das MFs do cluster estabelecida em um momento intermediário.

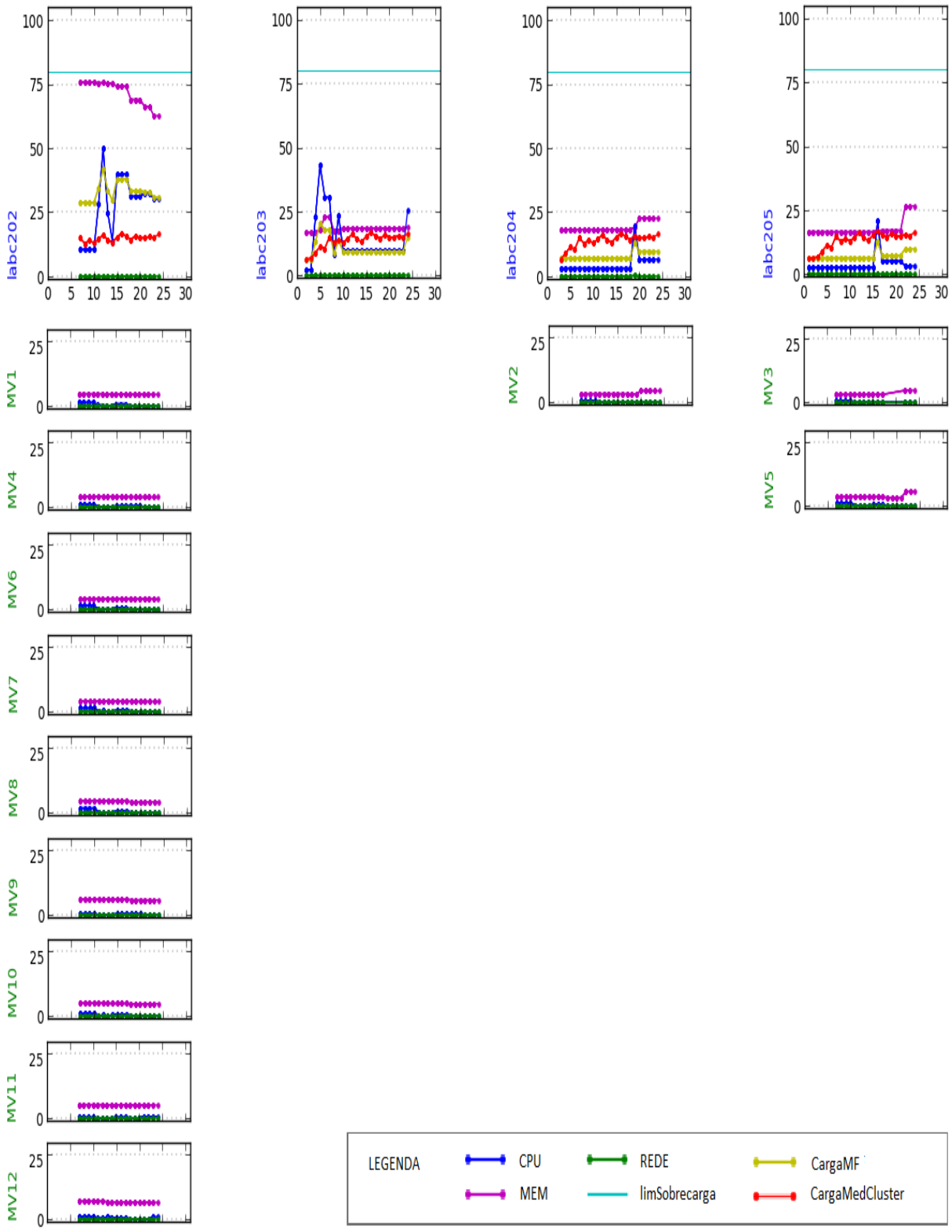

FIG. 5.12 Configuração Intermediária (1) utilizando a variação 10

Conforme pode ser visto na FIG. 5.12, a MF Lac202, após migrar 3 MVs, apresentou uma queda na utilização do recurso memória contribuindo para aproximação da sua CTMF com a CMC. Em relação à MF Labc204 e Labc205,

nota-se que estas, após receberem MVs, diminuíram o distanciamento de suas CTMFs em relação à CMC. Quanto a Labc203, nota-se que ela manteve o afastamento de sua CTMF em relação à CMC.

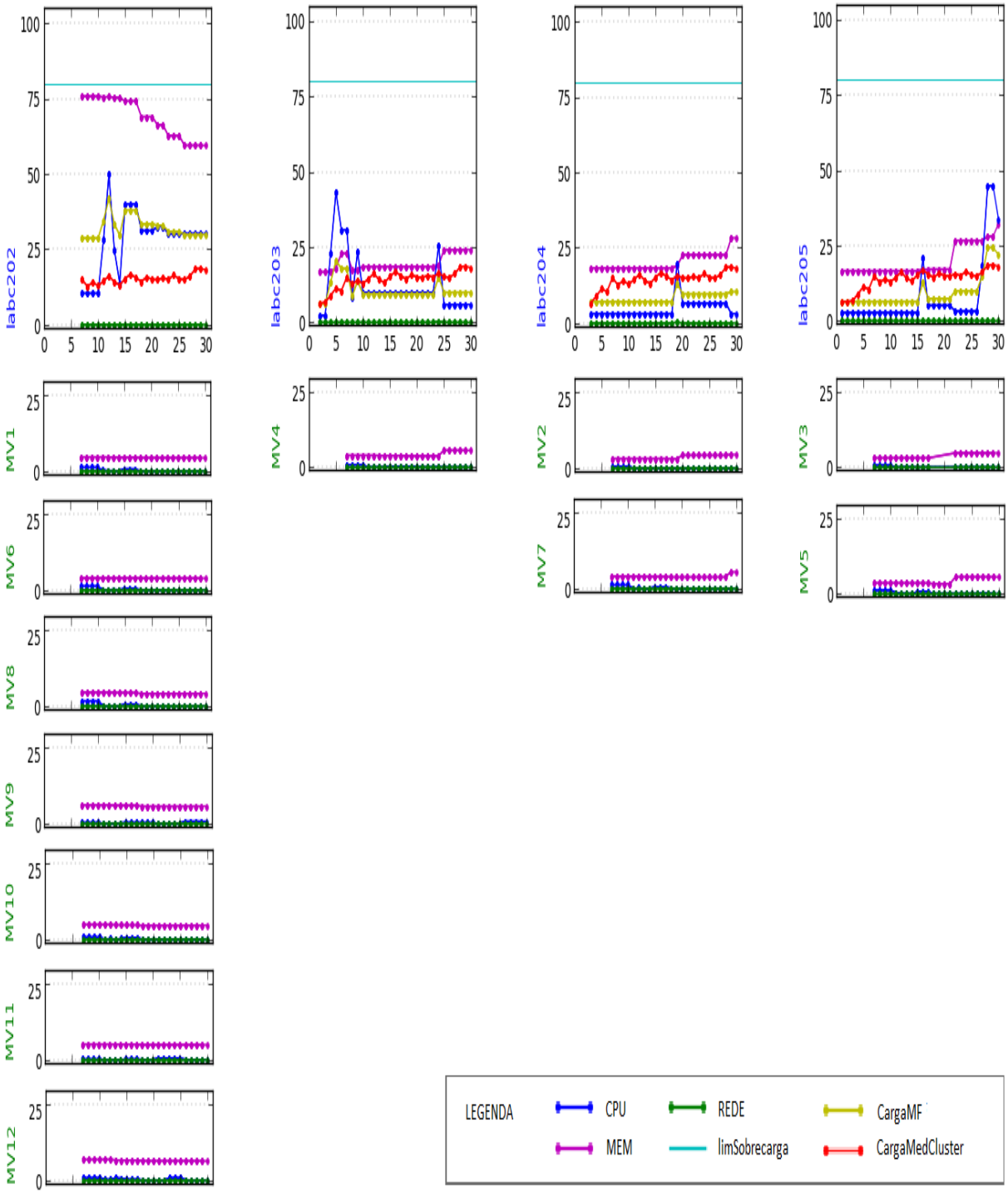

FIG. 5.13 Configuração Intermediária (2) utilizando a variação 10

Conforme pode ser visto na FIG. 5.13, a MF Lac202, após migrar 5 MVs, apresentou uma queda acentuada na utilização do recurso memória contribuindo para aproximação da sua CTMF com a CMC. Em relação às MF Labc203 e Labc204, nota-se que, após uma aproximação, mantiveram um pequeno distanciamento de suas CTMFs em relação à CMC. Quanto a Labc205, nota-se que, após a migração de 2 MVs, houve uma mudança no afastamento de sua CTMF em relação à CMC. A CTMF passou a figurar acima da CMC, porém dentro do afastamento previsto.

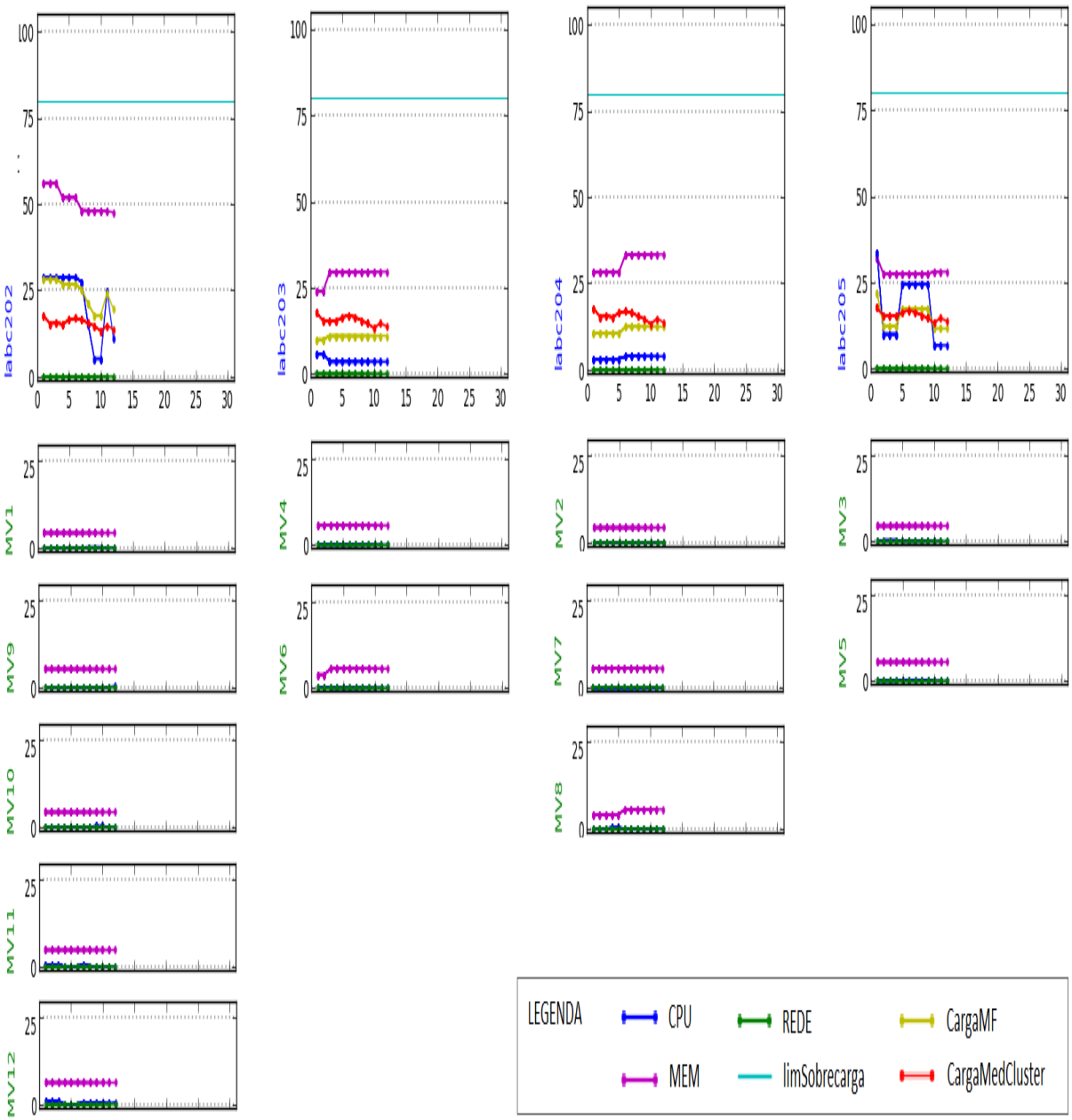

FIG. 5.14 Configuração Final do cluster utilizando a variação 10%

Após 7 migrações de MV, realizadas em 1 minuto e 10 segundos, a arquitetura finalizou o balanceamento de carga, conforme mostrado na FIG. 5.14. Observa-se que a Labc202, além de apresentar queda do recurso memória, manteve os demais recursos (CPU e rede) em condições compatíveis com as demais MFs do cluster. Houve também uma aproximação da sua CTMF com a CMC. Em relação às demais MFs do cluster, nota-se que todas alcançaram uma proximidade ainda maior de suas CTMF com a CMC, após as migrações realizadas. Todas as MFs encontram-se com a CTMF dentro do afastamento proposto. Assim, o cluster encontra-se balanceado.

5.3.5 RESULTADO DO EXPERIMENTO 1

Durante a distribuição das MVs pelas MFs do cluster por ocasião do balanceamento de carga, houve uma aproximação da CTMF com a CMC. Deste modo, as MFs passaram a ter suas CTMF dentro da Faixa de Equilíbrio, conforme o algoritmo proposto, em todos os afastamentos avaliados. Logo, o experimento 1 comprovou a efetividade da arquitetura para balanceamento do cluster.

Conforme pode ser visto na TAB. 5.6, a variação da CMC que apresentou a melhor distribuição das MVs pelo cluster (menor desvio padrão) com o menor número de migrações e menor tempo, foi a variação de 8 pontos percentuais. Por esse motivo, essa variação foi escolhida para ser utilizada como valor padrão para os demais experimentos deste trabalho. A distribuição das MVs no cluster foi utilizada somente como critério de desempate, pois como as MVs podem assumir tamanhos diferenciados de acordo com o ANS estabelecido, diminui a relevância da distribuição de MVs em relação aos cálculos efetuados para obtenção da CTMF, CMC. Estas refletem efetivamente a situação das MFs do cluster.

TAB. 5.6 Resumo do Experimento 1

Variação da Faixa de Equilíbrio Utilizada	Quantidade Média de Migrações	Tempo Médio Total Utilizado	Desvio Padrão do número de MVs por MF
3 pontos percentuais	13	3 mim e 5 segundos	1,2
5 pontos percentuais	8	2 mim e 38 segundos	0,8
8 pontos percentuais	8	1 mim e 19 segundos	0
10 pontos percentuais	7	1 mim e 10 segundos	1,4

O algoritmo estabelecido para a Phoenix permite que Faixa de Equilíbrio possa ser modificada a qualquer momento. Isto flexibiliza a distribuição das MVs, conforme a situação do cluster e vontade do administrador.

5.4 EXPERIMENTO 2 – TRATAMENTO DE SOBRECARGA EM MF

Este experimento tem por finalidade verificar a efetividade da arquitetura no trato de eventos de sobrecarga dos recursos (CPU, memória, rede) das MF, tendo como referência os ANSs dos serviços de infraestrutura (IaaS) hospedados.

As MFs hospedeiras de infraestrutura como serviço (*IaaS*), quando acometidas por sobrecargas momentâneas, podem ficar impedidas de realizar a entrega dos recursos (CPU, memória e rede) às MVs e, por sua vez, podem comprometer os ANSs estabelecidos.

Cada MV representa um serviço de infraestrutura, configurado de acordo com o ANS estabelecido, e esta se utiliza de recursos provenientes da MF hospedeira. Caso as MVs demandem recursos a ponto de atingir o limite de sobrecarga da MF, esta provavelmente começará a degradar os serviços hospedados. Deste modo, é fundamental que a arquitetura retire MVs da MF hospedeira a medida que o limiar de sobrecarga for atingido. Ao movimentar as MVs, é observada a situação geral de cluster.

Serão consideradas duas causas comuns de sobrecarga momentâneas em MFs: sobrecarga causada por *oversubscription* ou reserva de recursos acima da real capacidade da MF e sobrecarga espontânea ou sobrecarga gerada por algum processo interno do servidor por exemplo atualização de software.

No tratamento da sobrecarga da MF envolvendo a CPU, a memória e a rede, a arquitetura executa procedimentos semelhantes. Deste modo, o experimento considerou somente: (1) sobrecarga de CPU provocada espontaneamente na MF e (2) *Oversubscripiton* de memória provocado a partir de algumas MVs. Assim, outras variações de sobrecarga de MF envolvendo CPU, memória e rede não foram consideradas, pois podem ter seu comportamento inferidos a partir destes dois experimentos.

Incialmente, as MVs foram distribuídas de forma equilibrada no cluster de acordo com a Faixa de Equilíbrio, definida no Experimento 1. A partir desta situação, uma MF e algumas MVs foram submetidas às cargas geradas pelo programa Stress (AMOS WATERLAND, 2013).

Para identificar e tratar as sobrecargas em MFs e MVs, o algoritmo tomou como referência os limiares estabelecidos para arquitetura e definido no arquivo de configuração, descrito na Seção 4.4.

A TAB. 5.7 apresenta a configuração dos parâmetros utilizados para este experimento. Assim, o cluster foi configurado com 12 MVs distribuídas entre as MF do cluster de modo que a diferença da Carga da Total da MF (CTMF) e a CMC não ultrapassasse 8 pontos percentuais. Definida esta situação, a arquitetura apresentou os resultados descritos como se segue

TAB. 5.7 Parâmetros para realização do Experimento 2

Parâmetro	**Descrição**	**Valores Aceitos**	**Exemplo**
ordemMigraçãoMV	Indica a ordem na qual a arquitetura irá migrar a MVs	maiorCusto/menorCusto	ordemMigracaoMV= menorCusto
variacaoCMC	Variação permitida em relação à CMC	1 a 100	variacaoCMC = 8

5.4.1 SOBRECARGA DE CPU PROVOCADA NA MF

Uma vez o cluster balanceado, uma MF foi submetida a sobrecarga de CPU por meio do programa Stress (AMOS WATERLAND, 2013). Ao identificar que o limite de CPU da MF foi ultrapassado, o Monitor informa a sobrecarga (OVERLOADPM) ao Analisador que, após calcular a CMC, avalia a situação do cluster e determina a migração (MIGRATE) da MV de menor custo para MF

menos carregada, com menor CTMF, que possa receber a MV sem ficar sobrecarregada. Esta ação teve seu início, conforme mostrado na FIG. 5.15.

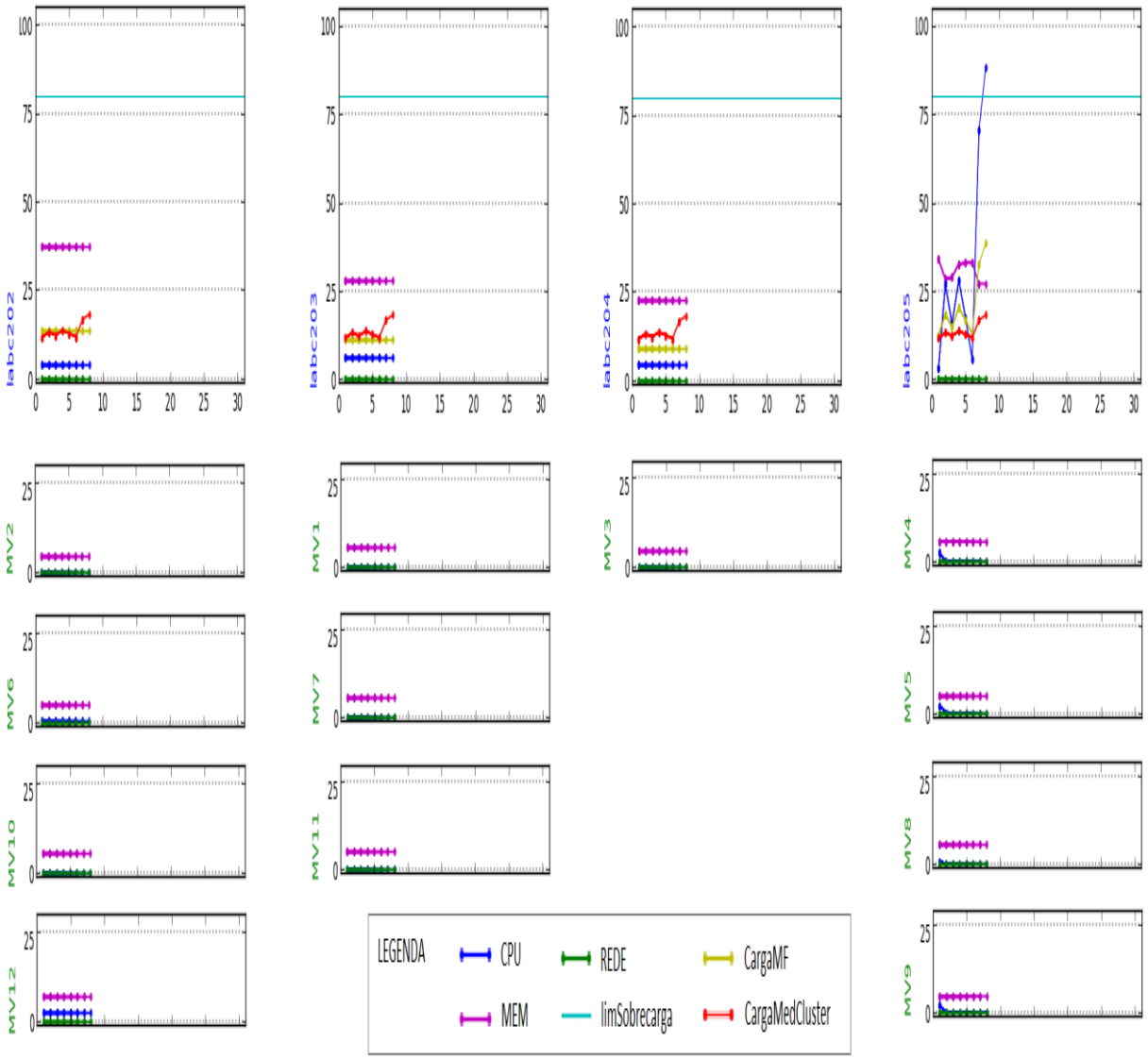

FIG. 5.15 Sobrecarga da CPU da MF - Configuração Inicial do Cluster

Uma vez identificada a causa da sobrecarga, o algoritmo determina que a MF Labc205 comece a retirar as MVs, de modo a preservar o ANS dos serviços hospedados. A FIG. 5.16 retrata a distribuição de MVs em um momento intermediário, após 2 MVs (MV5, MV9) terem sido migradas.

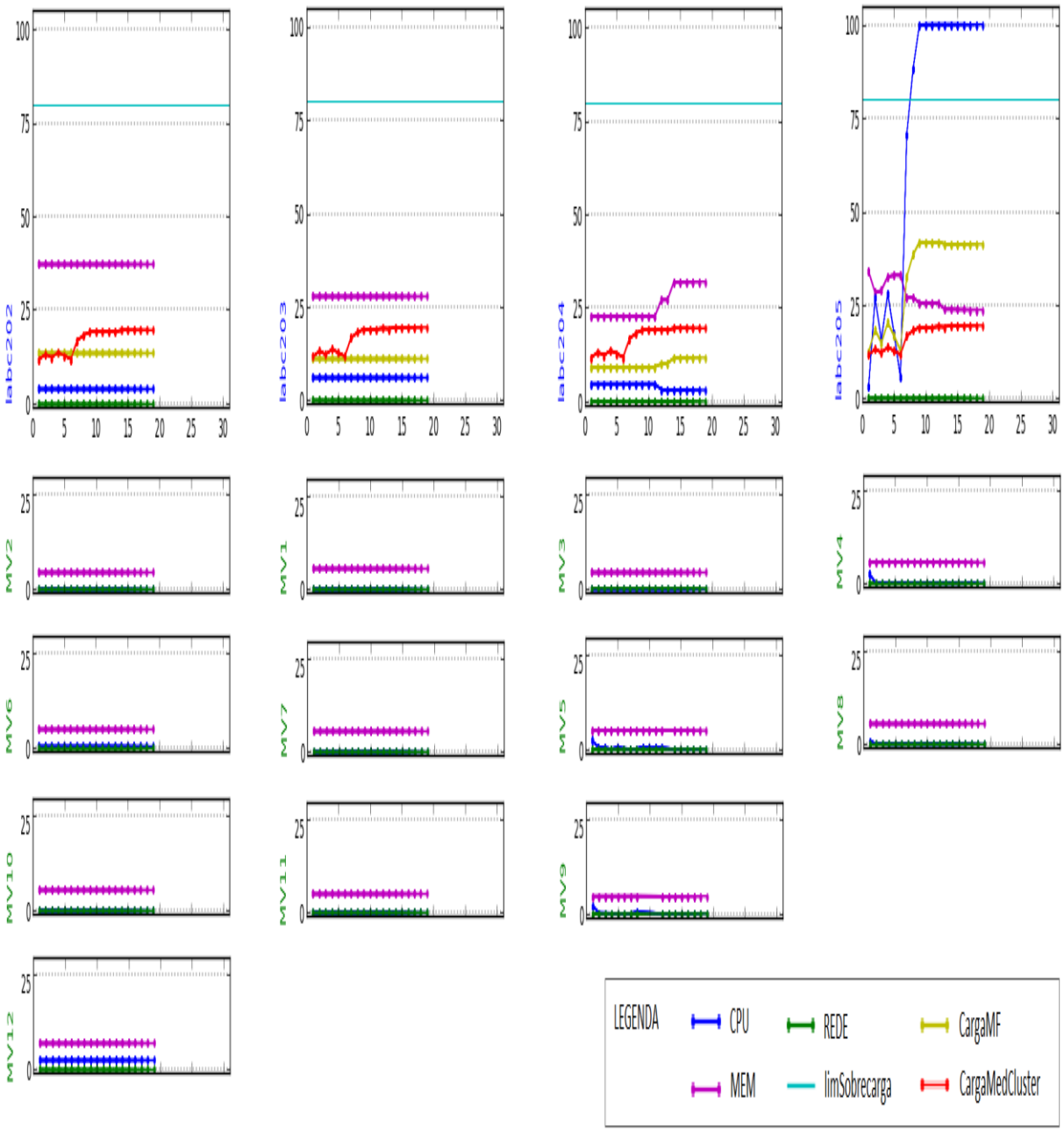

FIG. 5.16 Sobrecarga da CPU da MF - Configuração Intermediária (1) do Cluster

Após ter migrado todas as MVs e distribuído a carga entre as MFs do cluster, o algoritmo, depois de receber a confirmação de todas a migrações através de mensagens (MIGRATION_FINISHED e MONITOR_INFO), mantém a Labc205 vazia até que a MF volte a situação normal de funcionamento. Esta situação pode ser vista na FIG. 5.17.

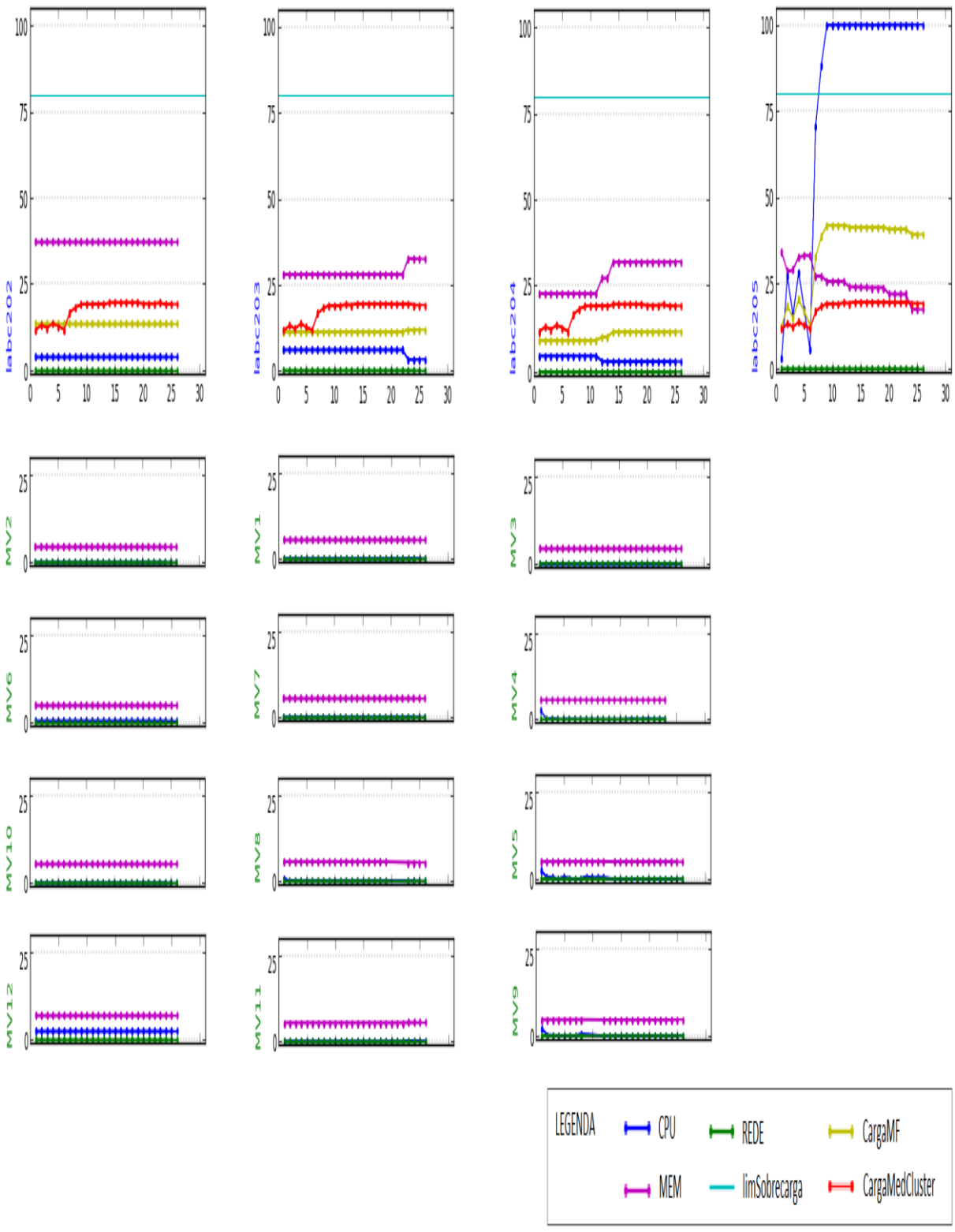

FIG. 5.17 Sobrecarga da CPU da MF - Configuração Final do Cluster

5.4.2 *OVERSUBSCRIPITON* NA MF PROVOCADO A PARTIR DAS MVS

Iniciando com o cluster balanceado, 3 MVs foram submetidas a sobrecarga de CPU por meio do programa Stress (AMOS WATERLAND, 2013). Ao requisitar o recurso CPU que tem direito, as MV passam a gerar uma sobrecarga da MF, uma vez que esta não tem recursos suficientes para atender toda a demanda.

Observa-se na FIG. 5.18 que as MVs (MV1, MV2, MV3) passaram a exigir da MF (Labc202) a capacidade prevista em ANS do recurso CPU. Ao fornecer a capacidade demandada, a MF (Labc202) tem seu recurso CPU intensamente solicitado.

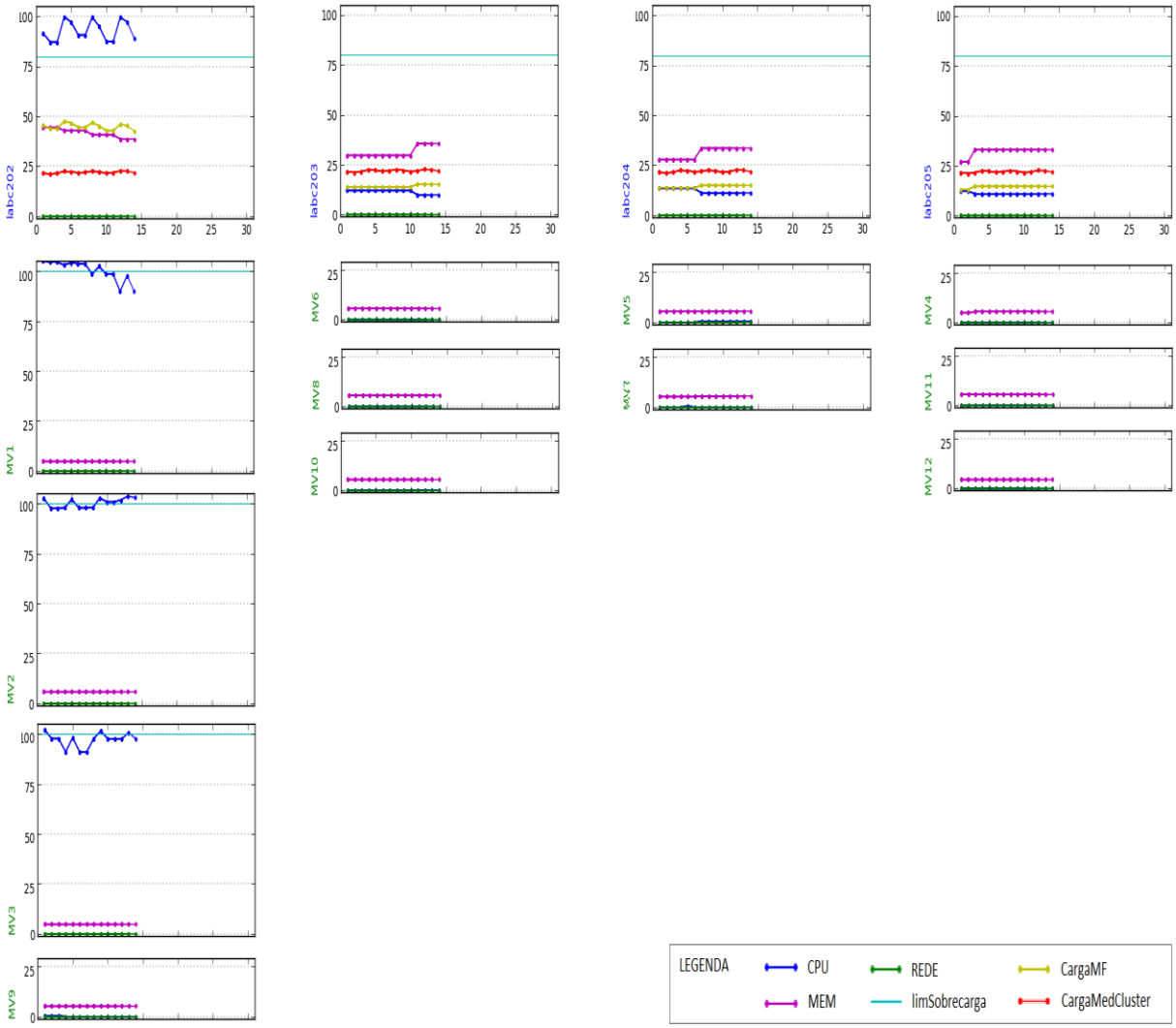

FIG. 5.18 Sobrecarga da CPU das MVs - Configuração Inicial do Cluster

Ao detectar a sobrecarga da MF, o módulo Monitor informa o evento relevante ao Analisador por meio de uma mensagem de MF sobrecarregada (OVERLOADPM). O Analisador, após receber a mensagem, calcula a CMC, avalia a situação do cluster e determina a migração (MIGRATE) da MV (MV3) que possui o menor custo para a MF menos carregada, MF (Labc204) com menor CTMF, que possa receber a MV sem ficar sobrecarregada. A MF (Labc202) que enviou a MV (MV3) informa o fim da migração (MIGRATION_FINISHED) e a MF (Labc204) recebedora informa sua nova configuração (MOINTOR_INFO). O resultado dessas ações pode ser visto na FIG. 5.19.

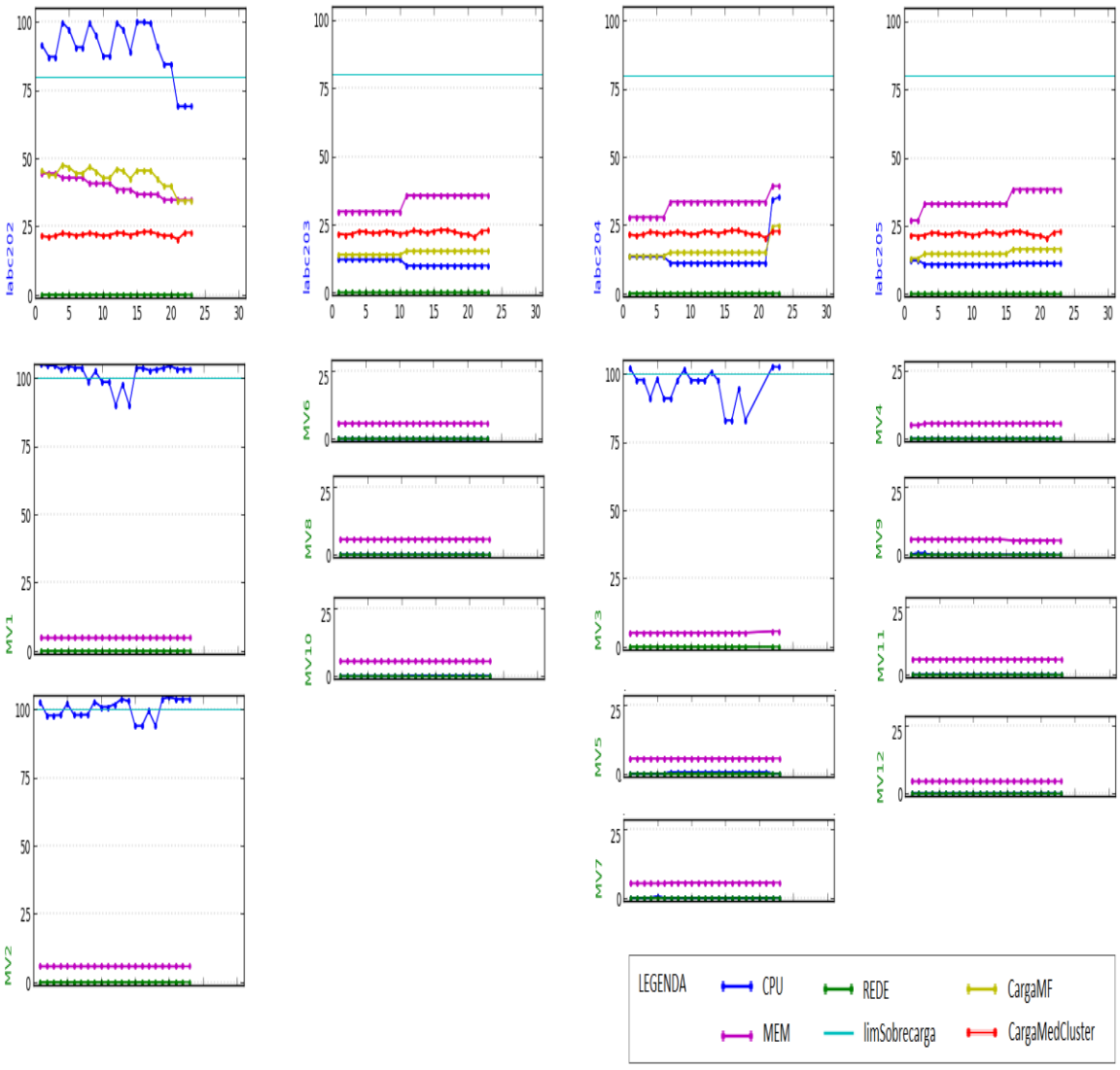

FIG. 5.19 Sobrecarga da CPU das MVs - Configuração Intermediária (1) do Cluster

Conforme pode ser visto na FIG. 5.19 após ter a sobrecarga tratada, a Labc202 voltou a sua condição normal de funcionamento. Porém, ao receber as informações (MONITOR_INFO) da MF (Labc204), o algoritmo identificou que há necessidade fazer o balanceamento de carga e então determina a migração de MVs de modo a balancear o cluster, conforme realizado no Experimento 1. A FIG. 5.20 retrata a distribuição final das MFs e MVs do cluster onde é possível perceber que o cluster encontra-se novamente balanceado.

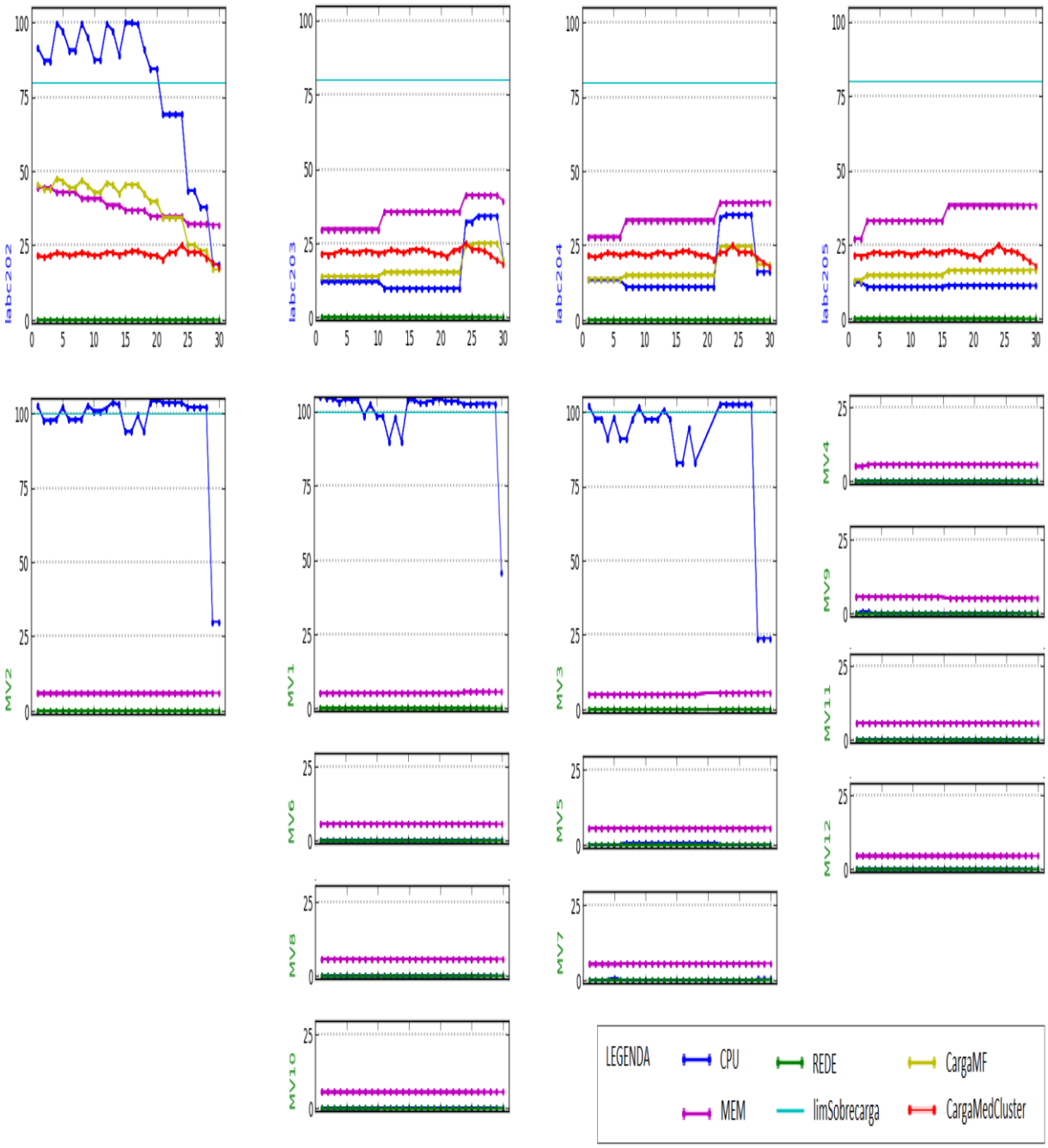

FIG. 5.20 Sobrecarga da CPU das MVs - Configuração Final do Cluster

5.4.3 RESULTADO DO EXPERIMENTO 2

Após esta rodada de experimentos, pode-se concluir que a arquitetura conseguiu tratar eventos de sobrecarga da MF que afetem os recursos (CPU e Memória) da MF, tendo como referência os ANSs.

O experimento avaliou o comportamento da arquitetura quando submetida a sobrecarga espontâneas e ao *oversubscription*, causado por demandas das MVs, demonstrando assim a efetividade do algoritmo para tratamento de sobrecarga da MF.

Conforme foi dito no início do experimento, as considerações aqui feitas em relação ao recurso CPU e memória também são válidas para variações do recurso rede.

5.5 EXPERIMENTO 3 – TRATAMENTO DE SOBRECARGA EM MV

Este experimento tem por finalidade verificar a efetividade da arquitetura Phoenix no tratamento da sobrecarga do recurso rede definido para cada MV, tendo como referência os ANSs dos serviços de infraestrutura (IaaS) hospedados.

Cada MV representa um serviço de infraestrutura hospedado. Para cada MV são configurados os recursos CPU, memória e rede, conforme descrito nos respectivos ANSs, e configurados conforme a TAB 5.1.

Os recursos CPU e memória e rede tem suas demandas controladas pelo hipervisor. Porém, o recurso rede sofre grande influência das requisições externas e caso haja um número de requisições acima do previsto para o serviço, mesmo que exista uma restrição do recurso rede configurado no MMV para o serviço, as requisições excedentes serão tratadas pela MF, podendo comprometer a disponibilização do recurso para as demais MVs hospedadas, ou mesmo causar a quebra do ANS estabelecido.

A importância deste experimento se dá porque o recurso rede é compartilhado por meio da utilização da placa de rede de MF. Sua utilização pode ser definida por meio da distribuição de um percentual da capacidade total recurso rede da MF e caso este percentual de rede configurado ultrapasse o limite, além de comprometer seu próprio serviço, pode comprometer os serviços

oferecidos pelas demais MVs que compartilham o recurso. O fato da demanda ter uma origem externa a MV e poder ser um *flash crowd*, por exemplo, exige cuidados adicionais, conforme o descrito na Seção 4.2.2.2.

Como as arquiteturas da referência não trataram efetivamente deste aspecto do tratamento da sobrecarga, surge, então, uma oportunidade de contribuição para este trabalho. Adicionalmente, cabe ressaltar que, para este experimento, foram consideradas requisições legítimas. Assim, estas devem ser tratadas conforme sejam demandadas.

De acordo com o descrito na Seção 4.2.2.2, de modo a evitar que sobrecargas momentâneas na rede das MVs comprometam os ANSs estabelecidos para os demais serviços, a arquitetura estabelece o isolamento da(s) MV(s) afetadas na MF hospedeira. As demais MVs devem ser migradas para não serem prejudicadas pela sobrecarga da MV afetada.

Para comprovar a efetividade do algoritmo, foram realizados vários experimentos, sendo que foram selecionadas três situações: a primeira mostra o funcionamento da Phoenix, conforme o comportamento proposto. A segunda e a terceira mostram a arquitetura com comportamento diferenciado, inspirado nas arquiteturas VOLTAIC (CARVALHO, 2012) e Sandpiper (WOOD et al., 2009).

Para realização deste experimento, o cluster foi configurado com 12 MVs distribuídas entre as MFs de modo que a diferença da carga total da MF (CTMF) e a CMC não ultrapassasse 8 pontos percentuais.

Para identificar e tratar as sobrecargas em MFs e MVs, o algoritmo tomou como referência os limiares estabelecidos para arquitetura e definido no arquivo de configuração, descrito na Seção 4.4.

Diferentemente do experimento 2, as sobrecargas não são tratadas como *oversubscrition.* Deste modo, partindo de um cluster balanceado será mostrado o comportamento da arquitetura diante da sobrecarga de rede em três situações:

- Tratamento de Sobrecarga de Rede MV, fixando a MV sobrecarregada na MF de origem e movendo primeiro as MVs de menor custo.
- Tratamento de Sobrecarga de Rede MV, não fixando a MV sobrecarregada na MF de origem e movendo primeiro as MVs de menor custo.

- Tratamento de Sobrecarga de Rede MV, não fixando a MV sobrecarregada na MF de origem e movendo primeiro as MVs de maior custo.

Diante das situações de sobrecarga descritas no item, a arquitetura apresentou os resultados descritos como se segue.

5.5.1 FIXAÇÃO DA MV SOBRECARREGADA

Este experimento mostra o comportamento da arquitetura quando submetida a sobrecarga de rede da MV, onde será fixada a MV afetada na MF hospedeira e migradas as MVs não afetas por ordem de menor custo.

Para caracterizar o uso dos parâmetros, estes foram colocados em destaque na TAB. 5.8.

TAB. 5.8 Parâmetros para realização do Experimento 3 (A)

Parâmetro	**Descrição**	**Valores Aceitos**	**Exemplo**
fixaMV	Indica que a arquitetura irá fixar a MV afetada pela sobrecarga da rede	SIM/NÃO	fixaMV= SIM
ordemMigraçaoMV	Indica a ordem na qual a arquitetura irá migrar a MVs	maiorCusto/menorCusto	ordemMigracaoMV= menorCusto
variacaoCMC	Variação permitida em relação à CMC	1 a 100	variacaoCMC = 8

Incialmente, as MVs foram distribuídas de forma equilibrada no cluster de acordo com a Faixa de Equilíbrio com variação de 8 pontos percentuais. A partir desta situação, uma MV (MV8) foi submetida às cargas geradas pelo programa iperf (IPERF, 2014).

Ao identificar que o limite de rede da MV foi ultrapassado, o Monitor informou ao Analisador tal situação (OVERLOADNETVM). Este, após calcular a CMC, avaliou a situação do cluster e determinou a migração (MIGRATE) das MVs de menor custo para a MF menos carregada, MF com menor CTMF, que pudesse receber a MV sem ficar sobrecarregada. Neste momento, a MV com sobrecarga de rede (MV8) foi fixada na MF hospedeira, enquanto persistisse a

sobrecarga da MV. Além de garantir mais recursos para a MV sobrecarregada, isto garantiu o isolamento do problema na MF hospedeira, não contaminando as demais MFs do cluster. Neste contexto, a distribuição das MVs é apresentada na FIG. 5.21. Nesta, pode ser identificado o aumento da carga da rede e da CPU tanto na MV quanto na MF, comprovando a tese que a MF é diretamente afetada pela utilização de recursos das MVs.

As requisições geradas pelo aplicativo iperf foram da ordem de 300 Mbp/s, sendo que o limite da MV era de 250 Mbps (a ferramenta poderia ter gerado até 990 Mbps somente com este gerador). Deste modo, a carga na MV ficou estacionada em 100% de sua capacidade (250 Mbps), enquanto que a MF continuou tratando o excesso de requisições.

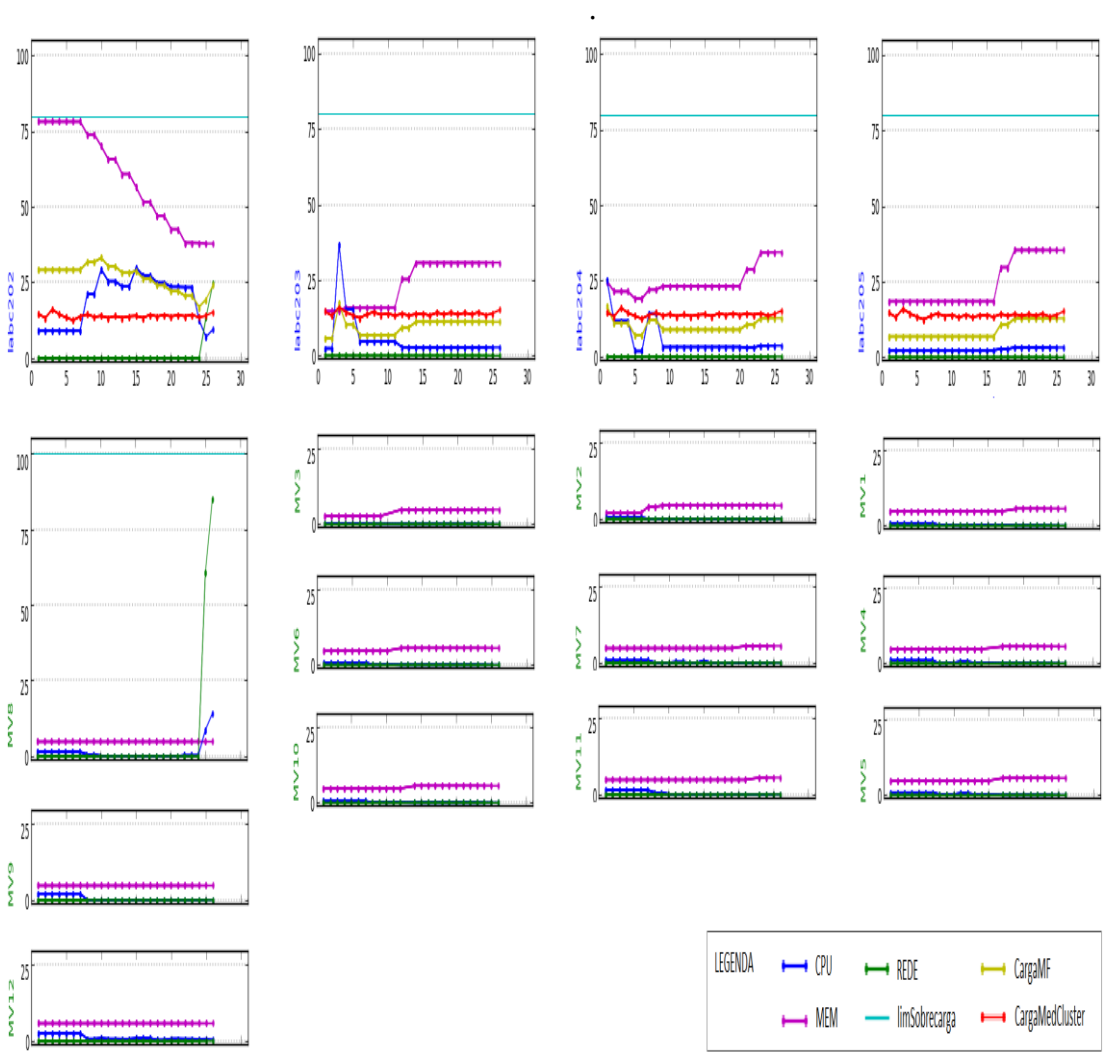

FIG. 5.21 Sobrecarga de Rede MV, fixando a MV Situação Inicial do Cluster

Uma vez identificada a causa da sobrecarga (OVERLOADNETVM), o algoritmo determinou que a MF Labc204 começasse a retirar a MVs, de modo a preservar o ANS dos serviços hospedados. A FIG. 5.22FIG. 5.21 retrata a distribuição de MVs em um momento intermediário, após uma MV (MV9) ter sido migrada.

Observa-se na FIG. 5.22 que, em razão da sobrecarga da rede, a CTMF da MF (Labc202) está acima das outras MFs.

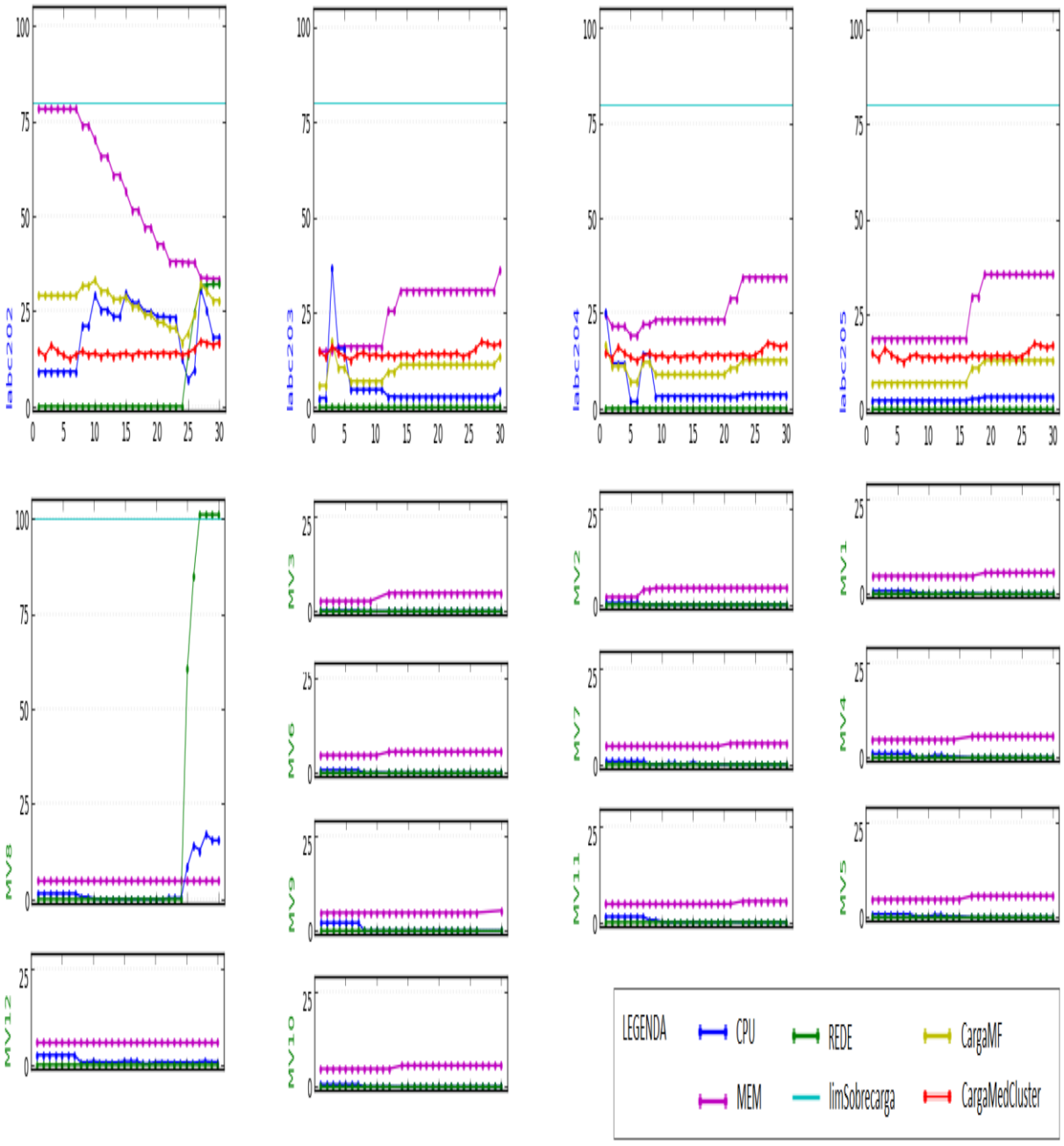

FIG. 5.22 Sobrecarga de Rede MV, fixando a MV -Situação intermediaria (1) do Cluster

Após ter migrado todas as MVs não acometidas pela sobrecarga de rede, ter fixado a MV sobrecarregada na MF original, ter distribuído a carga das MVs migradas entre as MFs do cluster e receber a confirmação de todas as migrações através de mensagens (MIGRATION_FINISHED e MONITOR_INFO), o algoritmo manteve a Labc202 somente com a MV8 hospedada, enquanto perdurou a situação de sobrecarga. Esta situação pode ser vista na FIG. 5.23

Cabe ressaltar que o cluster permaneceu equilibrado, mesmo tendo a MF Labc03 com o recurso memória um pouco mais elevado em razão da utilização do recurso pelas MVs hospedadas, conforme pode ser visto na FIG. 5.23.

.

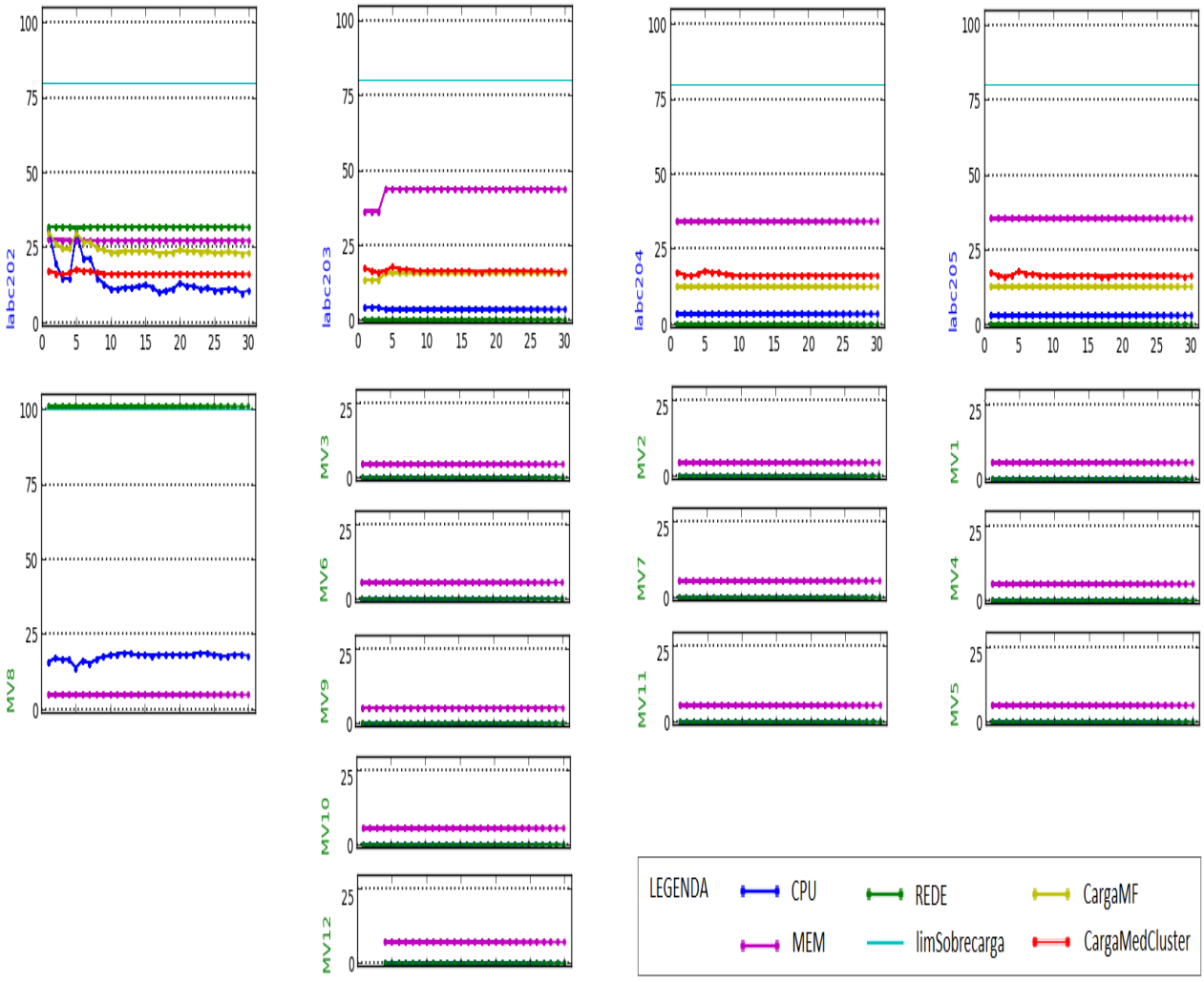

FIG. 5.23 Sobrecarga de Rede MV, fixando a MV - Situação Final do Cluster

É importante ressaltar que, mesmo tendo recebido mensagens e confirmação de migração da MF recebedora (MONITOR_INFO), o Analisador

priorizou o tratamento da sobrecarga, conforme apresentado na Seção 4.2.2.2, impedindo assim, que mesmo havendo capacidade da MF Labc02, MVs fossem realocadas nesta MF enquanto a situação de sobrecarga persistiu.

5.5.2 NÃO FIXAÇÃO DA MV SOBRECARREGADA, MIGRANDO MV DE MENOR CUSTO

Este experimento mostra o comportamento da arquitetura quando a MV é submetida a sobrecarga de rede, sendo que esta não é fixada MF hospedeira e as demais MVs são migradas por ordem de menor custo.

Para caracterizar o uso dos parâmetros, foram colocados em destaque na TAB. 5.9 os parâmetros da TAB. 4.5 modificados para realização deste experimento como segue.

TAB. 5.9 Parâmetros para Realização do Experimento 3 (B)

Parâmetro	**Descrição**	**Valores Aceitos**	**Exemplo**
FixaMV	Indica que a arquitetura irá fixar a MV afetada pela sobrecarga da rede	SIM/NÃO	FixaMV= NAO
ordemMigracaoMV	Indica a ordem na qual a arquitetura irá migrar as MVs	maiorCusto/menorCusto	ordemMigracaoMV= menorCusto
variacaoCMC	Variação permitida em relação à CMC	1 a 100	variacaoCMC = 8

Incialmente, as MVs foram distribuídas de forma equilibrada no cluster de acordo com a Faixa de Equilíbrio definida no Experimento 1. A partir desta situação, uma MV foi submetida à carga gerada pelo programa iperf (IPERF, 2014).

Ao identificar que o limite de rede da MV foi ultrapassado o Monitor informou ao Analisador tal situação (OVERLOADNETVM). Este, após calcular a CMC, avaliou a situação do cluster e determinou a migração (MIGRATE) das MVs de menor custo para a MF menos carregada, MF com menor CTMF, que pudesse receber a MV sem ficar sobrecarregada. A MV com sobrecarga de rede (MV8) não foi fixada na MF hospedeira, enquanto persistisse a sobrecarga da

MV. Deste modo, o problema não ficou isolado na MF hospedeira, possibilitando a contaminação das demais MFs do cluster. Esta ação teve seu início conforme mostrado na FIG. 5.24. Nesta, pode ser identificado o aumento da carga da rede e da CPU tanto na MV quanto na MF mais uma vez comprovando a tese que a MF é diretamente afetada pela utilização de recursos das MVs.

As requisições geradas pelo aplicativo iperf foram da ordem de 850 Mbp/s, sendo que o limite da MV era de 250 Mbps.

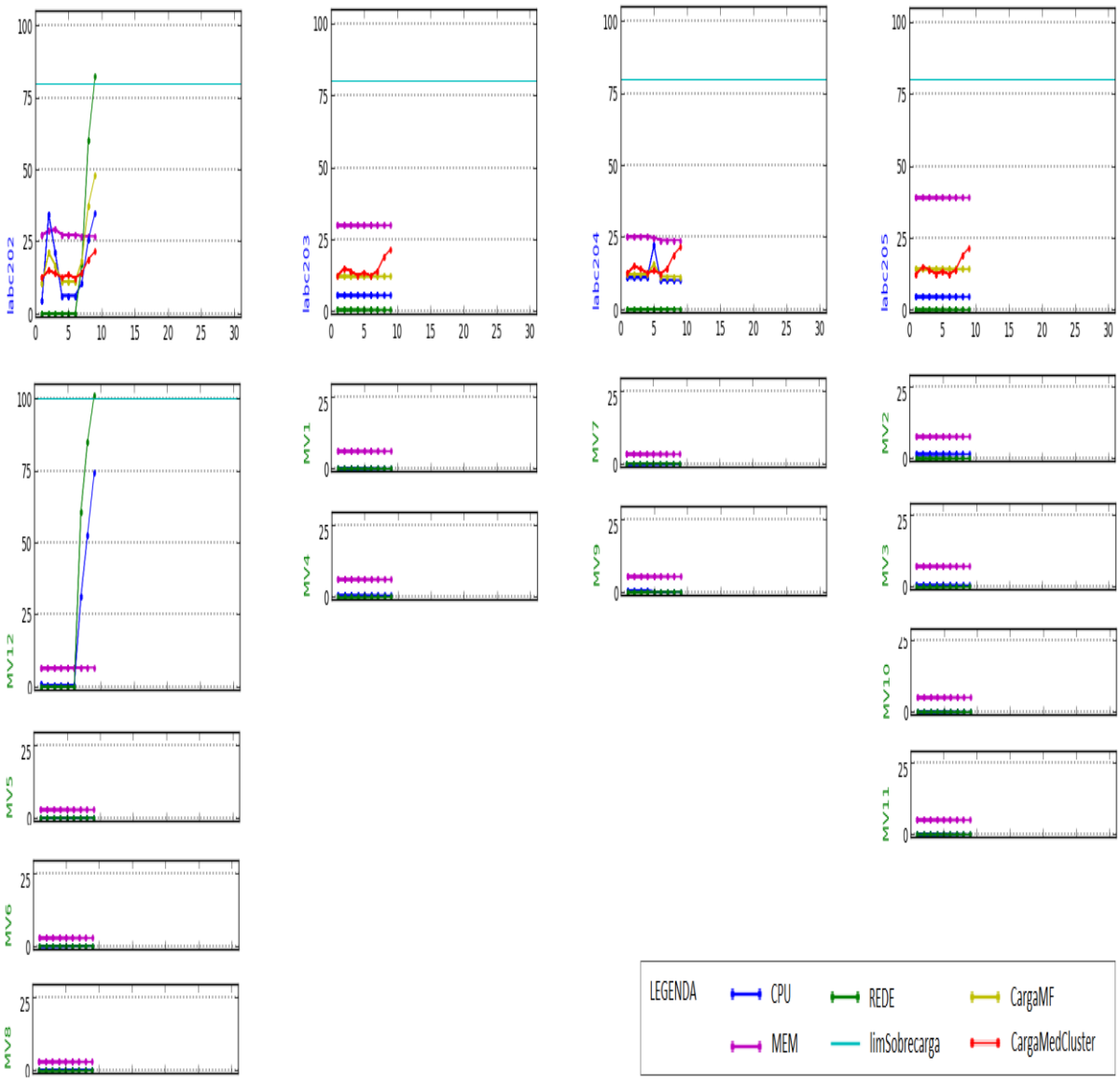

FIG. 5.24 Sobrecarga de Rede MV, sem Fixação da MV - Situação Inicial do Cluster

Uma vez identificada a causa da sobrecarga, o algoritmo determinou à MF Labc202 a retirada das MVs, de modo a preservar o ANS dos serviços

hospedados. A FIG. 5.25 retrata a distribuição de MVs após a migração de três MVs (MV6, MV8 e MV5).

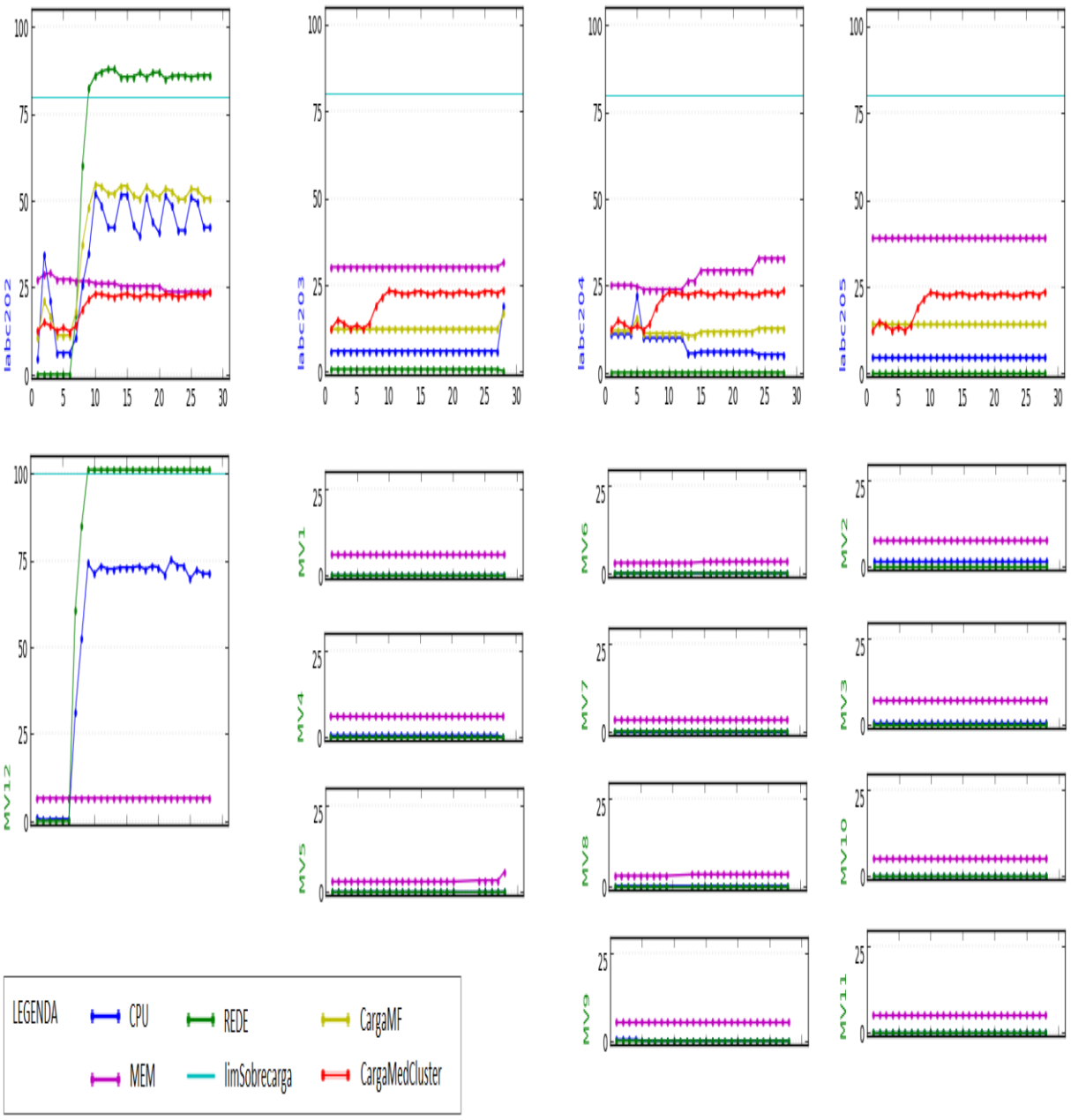

FIG. 5.25 Sobrecarga de Rede MV, sem Fixação da MV - Situação Intermediária (1) do Cluster

Uma vez migradas as MVs de menor custo, persistindo a sobrecarga de rede, o algoritmo determinou à MF Labc202 a retirada da MV 12, mesmo estando sobrecarregada. Com isso a CTMF da Lac202 diminuiu e a CTMF da Lac203, que era a menor, passou a subir. A FIG. 5.26 retrata a distribuição de MVs após a migração da MV12 (sobrecarregada).

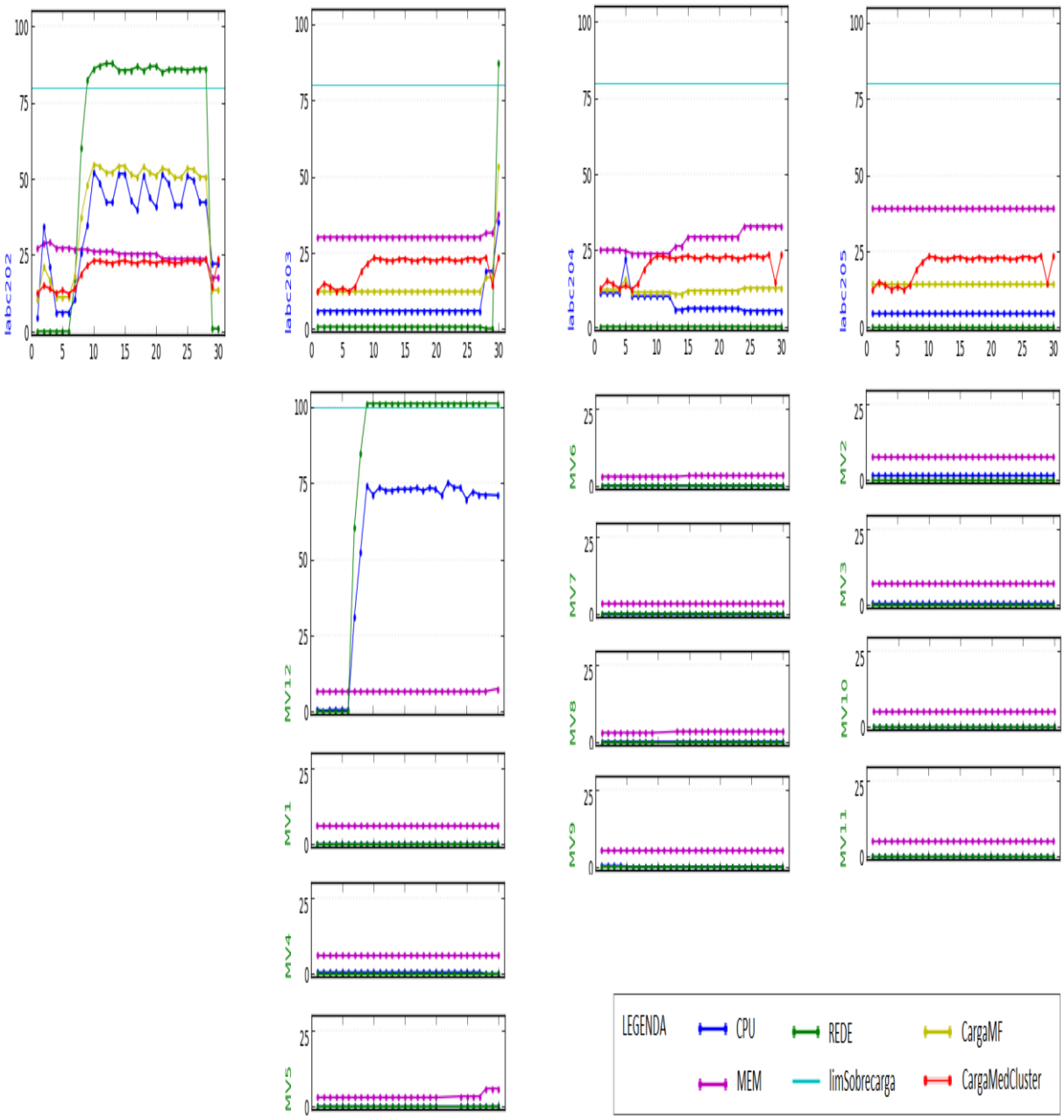

FIG. 5.26 Sobrecarga de Rede MV, sem Fixação da MV - Situação Intermediária (2) do Cluster

Conforme a FIG. 5.27, a MF Labc203 recebeu a MV12 com sobrecarga de rede e teve seus recursos (CPU, memória e rede) comprometidos, colocando em risco as MVs hospedadas.

Ao identificar a sobrecarga causada pela MV12, o Analisador determina a migração das MVs (MV1, MV4 e MV5) para a Labc202 (MF com menor CTMF).

Após esta operação, a MV 12 ficou sozinha na Labc203, porém, como não foi fixada, pode ser migrada para outra MF menos carregada.

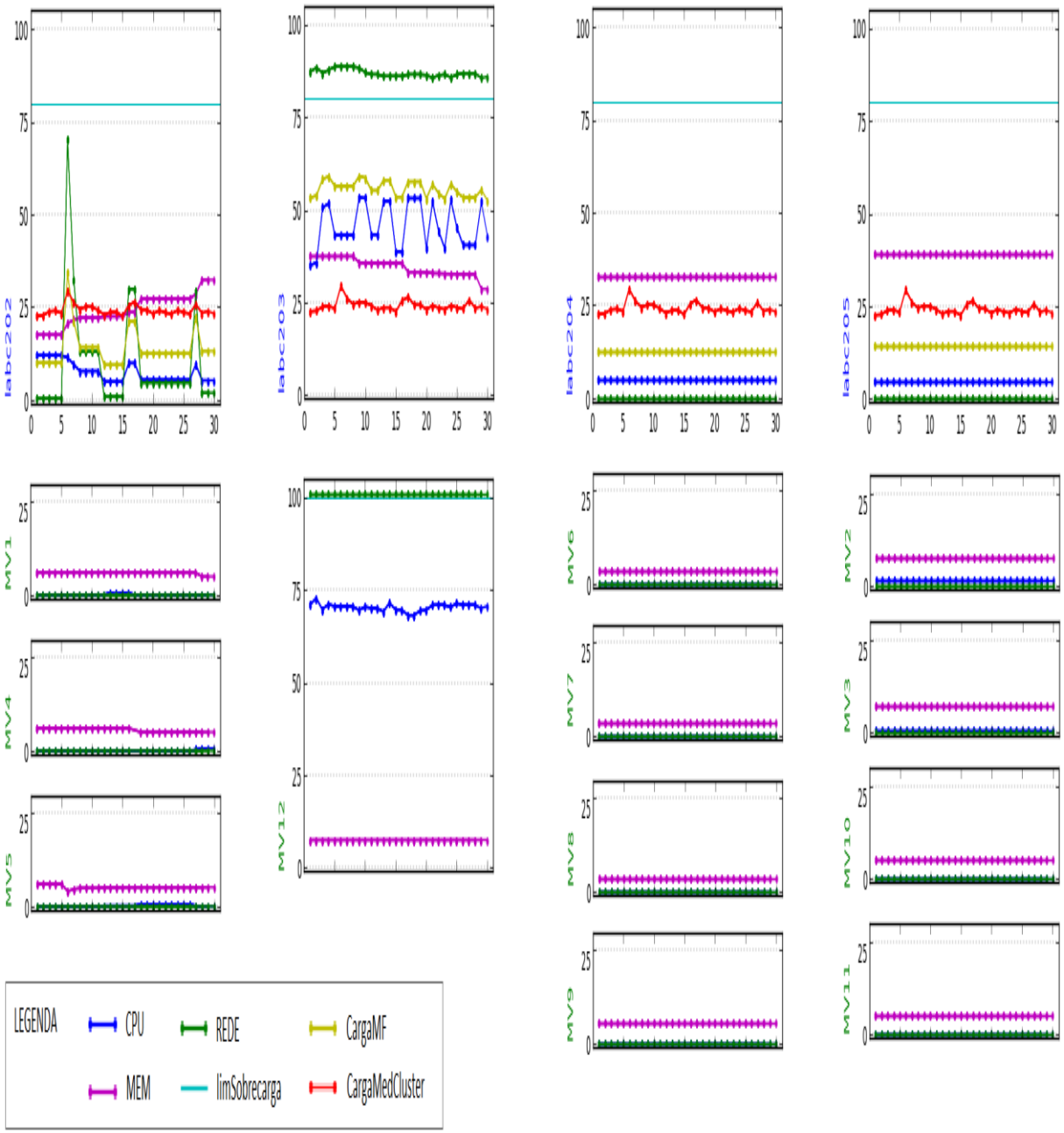

FIG. 5.27 Sobrecarga de Rede MV, sem Fixação da MV - Situação Intermediária (3) do Cluster

Conforme pode ser visto na FIG. 5.28 a MV12 foi migrada para LABC202 provocando novamente a sobrecarga do recurso rede e aumento da utilização dos recursos CPU e memória.

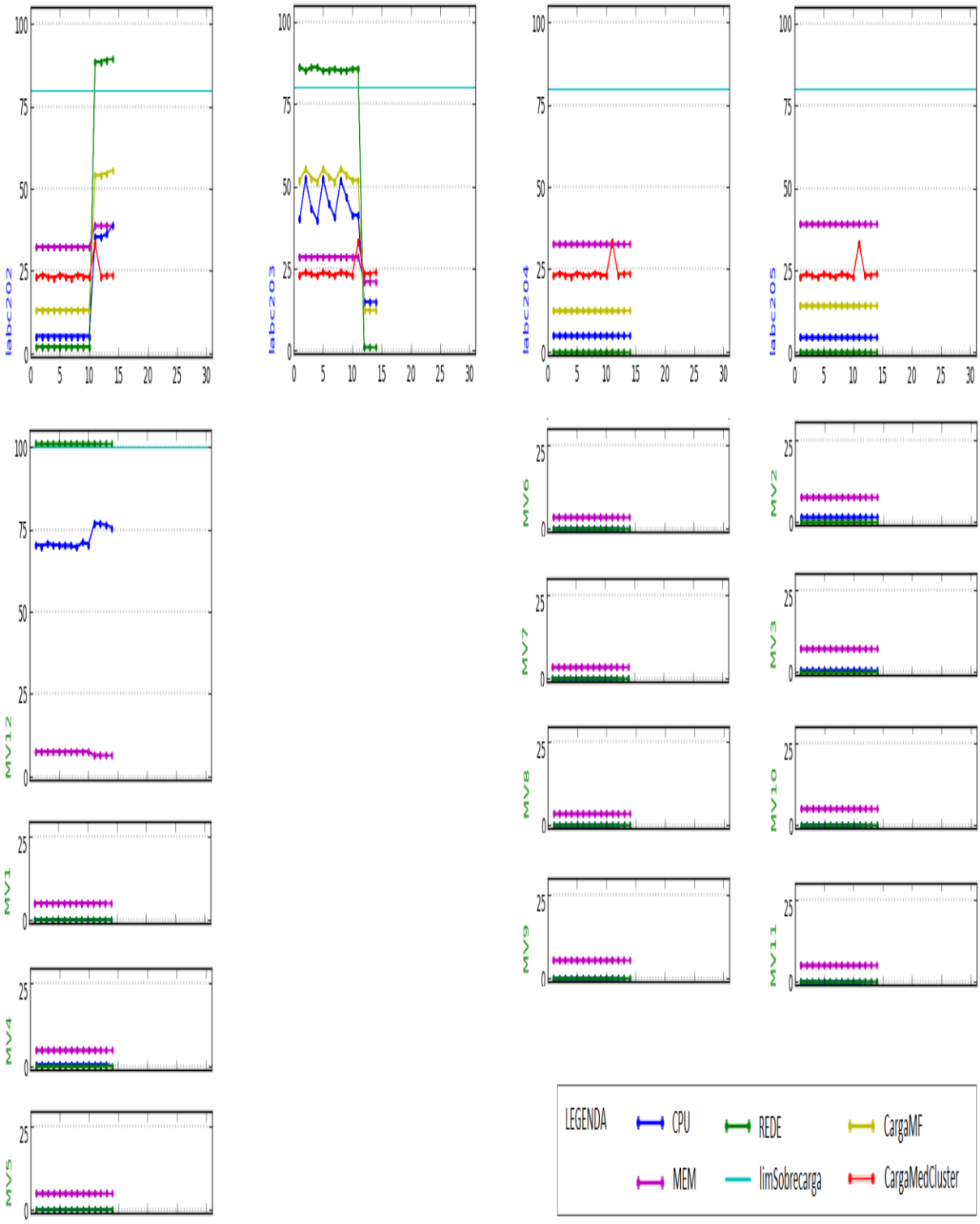

FIG. 5.28 Sobrecarga de Rede MV, sem Fixação da MV - Situação Final do Cluster

Conforme pode ser visto neste experimento, a não fixação da MV com sobrecarga de rede pode comprometer não só as MV que estiverem na mesma

MF, mas também outras MFs. Por tanto não bastar retirar as MVs de menor custo primeiro, é preciso isolar a sobrecarga para evitar a quebra de serviço de outras MVs ou a indisponibilidade de outras MFs, impedindo-as de oferecer os recursos de rede contratados para os serviços hospedados.

5.5.3 NÃO FIXAÇÃO DA MV SOBRECARREGADA E PRIORIZAÇÃO DA MV DE MAIOR CUSTO

Este experimento mostra o comportamento da arquitetura quando submetida a sobrecarga de rede da MV (OVERLOADNETMV), onde a MV afetada não foi fixada na MF hospedeira e as MVs não sobrecarregadas foram migradas por ordem maior de custo.

Este é parte do comportamento esperado para arquiteturas VOLTAIC (CARVALHO, 2012) e Sandpiper (WOOD et al., 2009), uma vez que a primeira migra MVs que necessitam de mais recursos de CPU e memória para MFs com capacidade para atender a demanda, enquanto a segunda migra MVs com comportamento anômalo para uma determinada MF, levando em consideração a utilização dos recursos CPU e memória.

Para caracterizar o uso dos parâmetros, foram colocados em destaque na TAB. 5.10, os parâmetros da TAB. 4.5 modificados para realização deste experimento como segue.

TAB. 5.10 Parâmetros para realização do Experimento 3 (C)

Parâmetro	**Descrição**	**Valores Aceitos**	**Exemplo**
Fixa_MV	Indica que a arquitetura irá fixar a MV afetada pela sobrecarga da rede	SIM/NÃO	Fixa_MV= NAO
ordemMigracaoMV	Indica a ordem na qual a arquitetura irá migrar as MVs	MaiorCusto/MenorCusto	ordemMigracaoMV= maiorCusto
variacaoCMC	Variação permitida em relação à CMC	1 a 100	variacaoCMC = 8

Inicialmente, as MVs foram distribuídas de forma equilibrada no cluster de acordo com a Faixa de Equilíbrio de 8 pontos percentuais. A partir desta situação, uma MV (MV1) foi submetida à carga gerada pelo programa iperf (IPERF, 2014).

As requisições geradas pelo aplicativo iperf foram da ordem de 870 Mbp/s, sendo que o limite da MV era de 250 Mbps.

Ao identificar que o limite de rede da MV foi ultrapassado, o Monitor informou ao Analisador tal situação (OVERLOADNETVM). Este, após calcular a CMC, avaliou a situação do cluster e determinou a migração (MIGRATE) das MVs de maior custo para a MF menos carregada, MF com menor CTMF, que pudesse receber a MV sem ficar sobrecarregada. A MV com sobrecarga de rede (MV1) não foi fixada na MF hospedeira, enquanto persistisse a sobrecarga da MV. Deste modo, o problema não ficou isolado na MF hospedeira, possibilitando a contaminação das demais MFs do cluster. Esta ação teve seu início, conforme mostrado na FIG. 5.29.

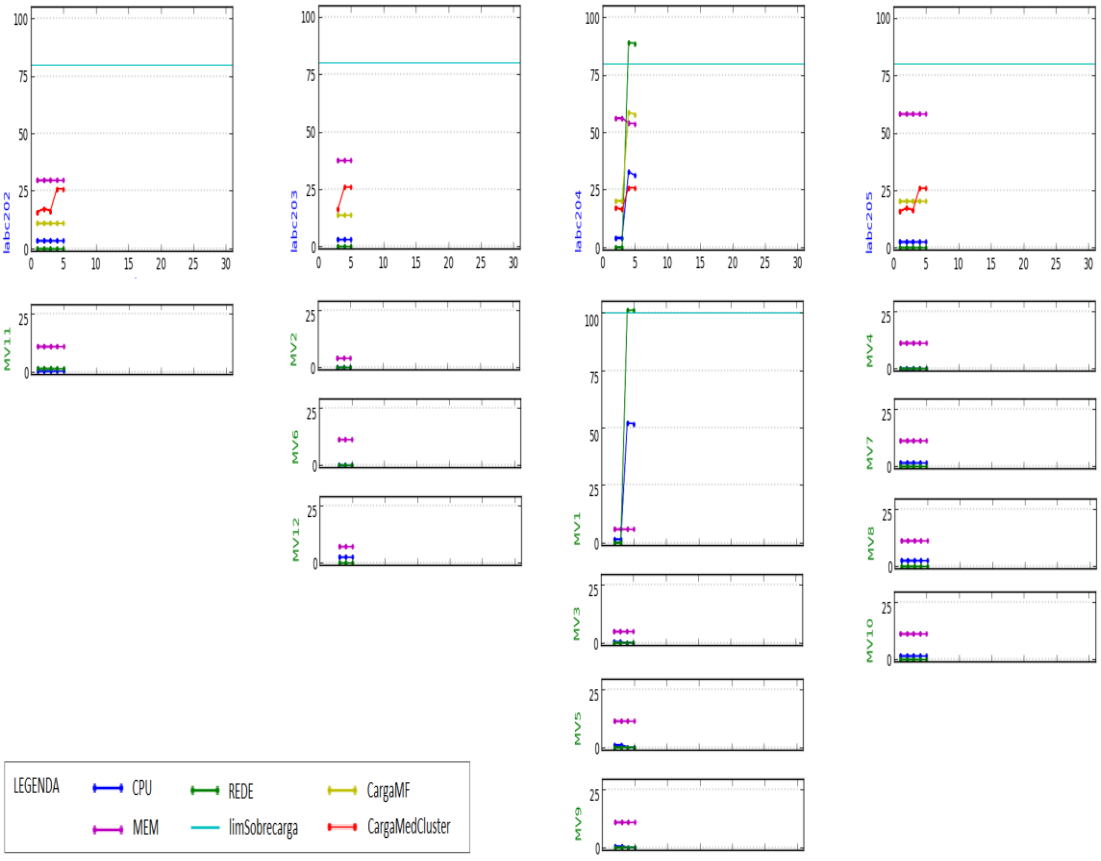

FIG. 5.29 Sobrecarga de Rede MV - Situação inicial do Cluster

Como resultado da identificação da sobrecarga, o algoritmo determina a migração da MV1 uma vez que esta apresentou maior custo (maior necessidade de recursos se levarmos em consideração o comportamento do Sanpider e VOLTAIC) em relação as outras MVs. Deste modo, o algoritmo garantiu mais recursos para MV1, sobrecarregada, mas o problema não ficou isolado na MF hospedeira (Labc204), e contaminou outra MF(Labc202) do cluster. A FIG. 5.30 retrata a distribuição de MVs após a MV1 ter sido migrada.

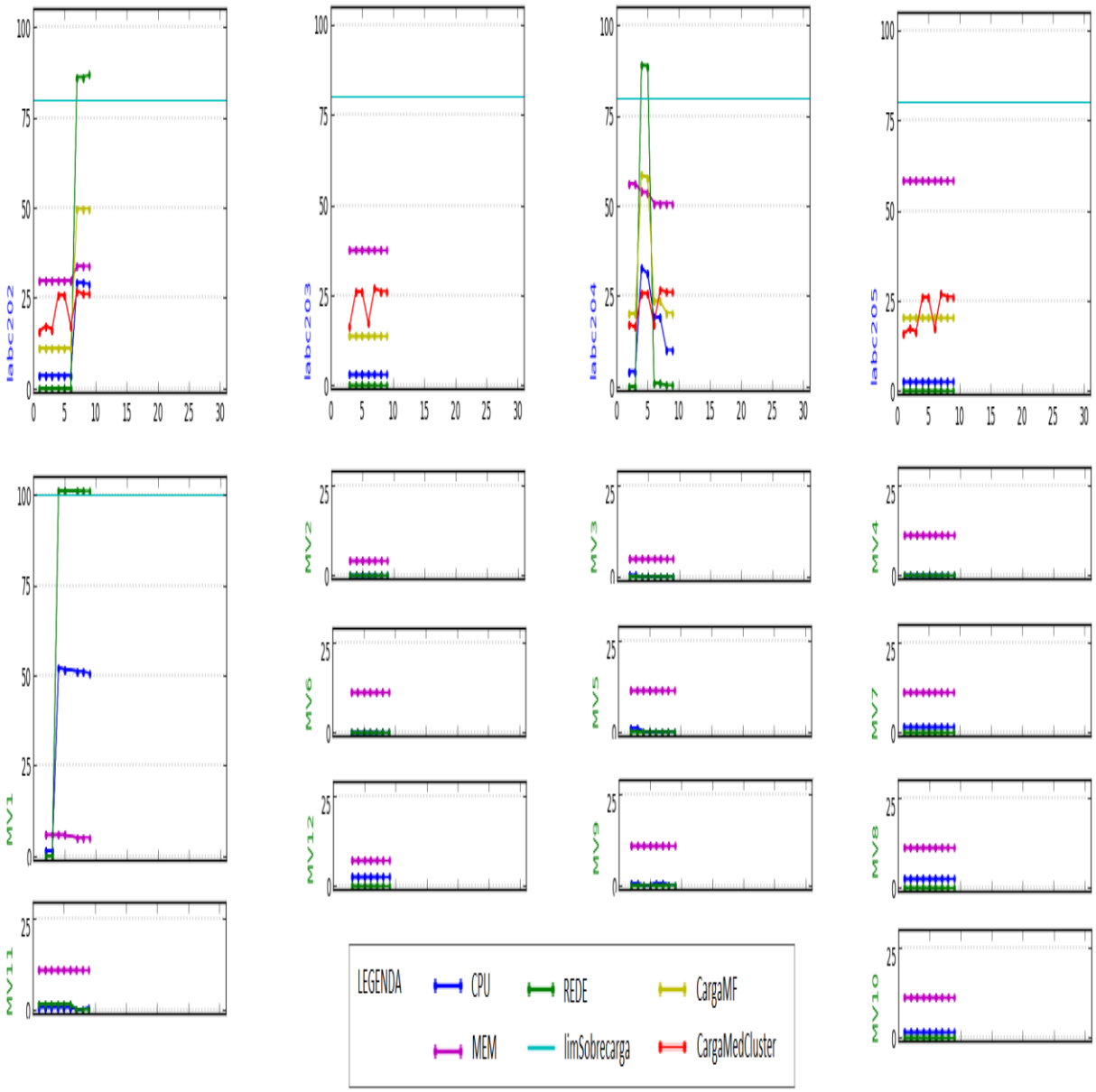

FIG. 5.30 Sobrecarga de Rede MV - Situação Intermediária (1) do Cluster

Uma vez migrada a MV de maior custo para a Labc202, persistindo a sobrecarga de rede, o algoritmo determinou à MF Labc202 a retirada da MV1, mesmo estando sobrecarregada. Com isso a CTMF da Lac202 diminuiu e a CTMF da Lac203, que era a menor, passou a subir. A FIG. 5.31 retrata a distribuição de MVs após a segunda migração da MV1.

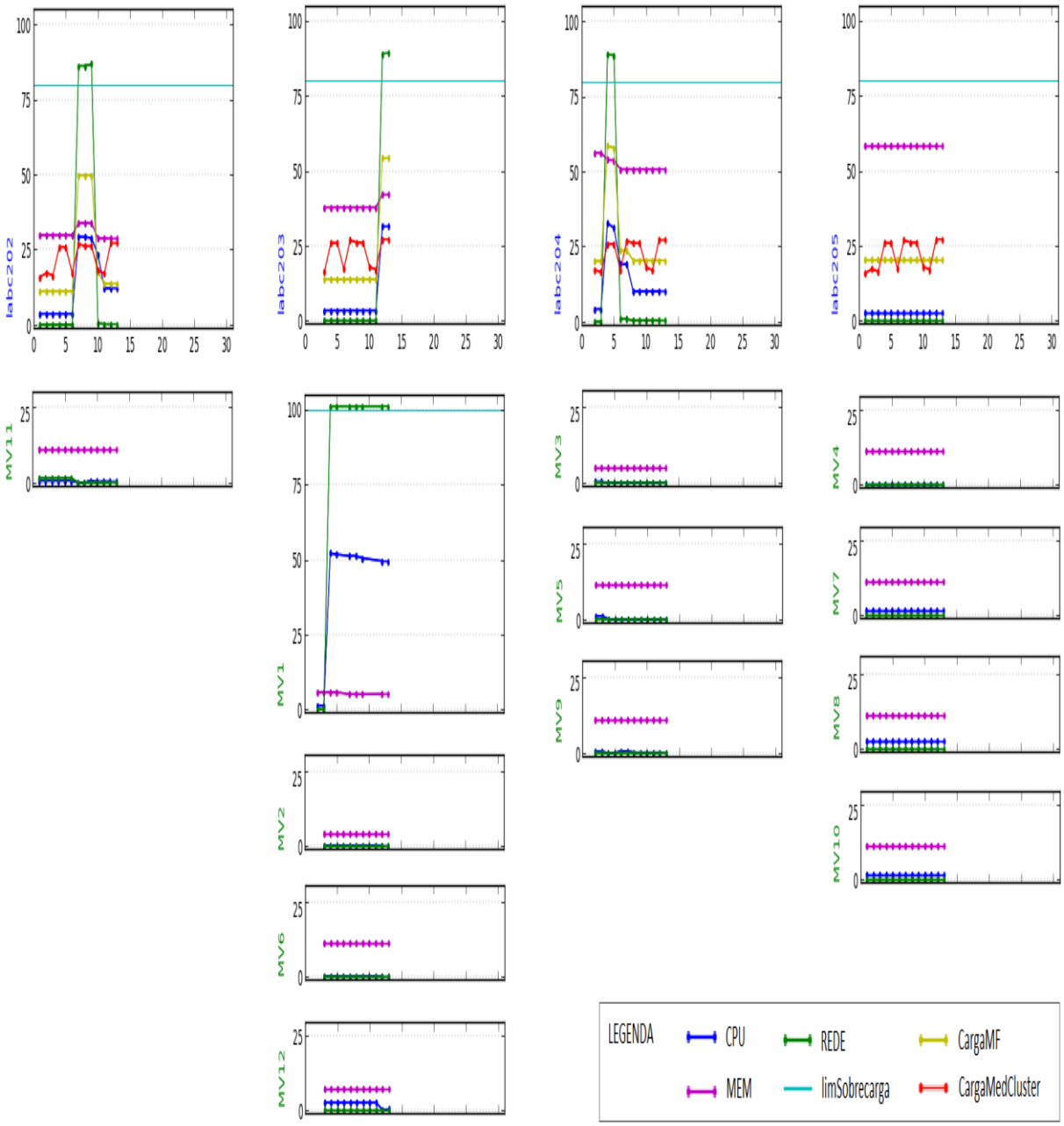

FIG. 5.31 Sobrecarga de Rede MV - Situação Intermediária (2) do Cluster

Uma vez migrada a MV1, persistindo a sobrecarga de rede, o algoritmo busca outra MF que tenha condições de atender a demanda da MV. Determinou então à MF Labc202 a retirada da MV1, mesmo estando sobrecarregada. Com isso, a CTMF da Lac202 diminuiu e a CTMF da Lac204, que era a menor, passou a subir. A FIG. 5.32 retrata a distribuição de MVs após a migração da MV1.

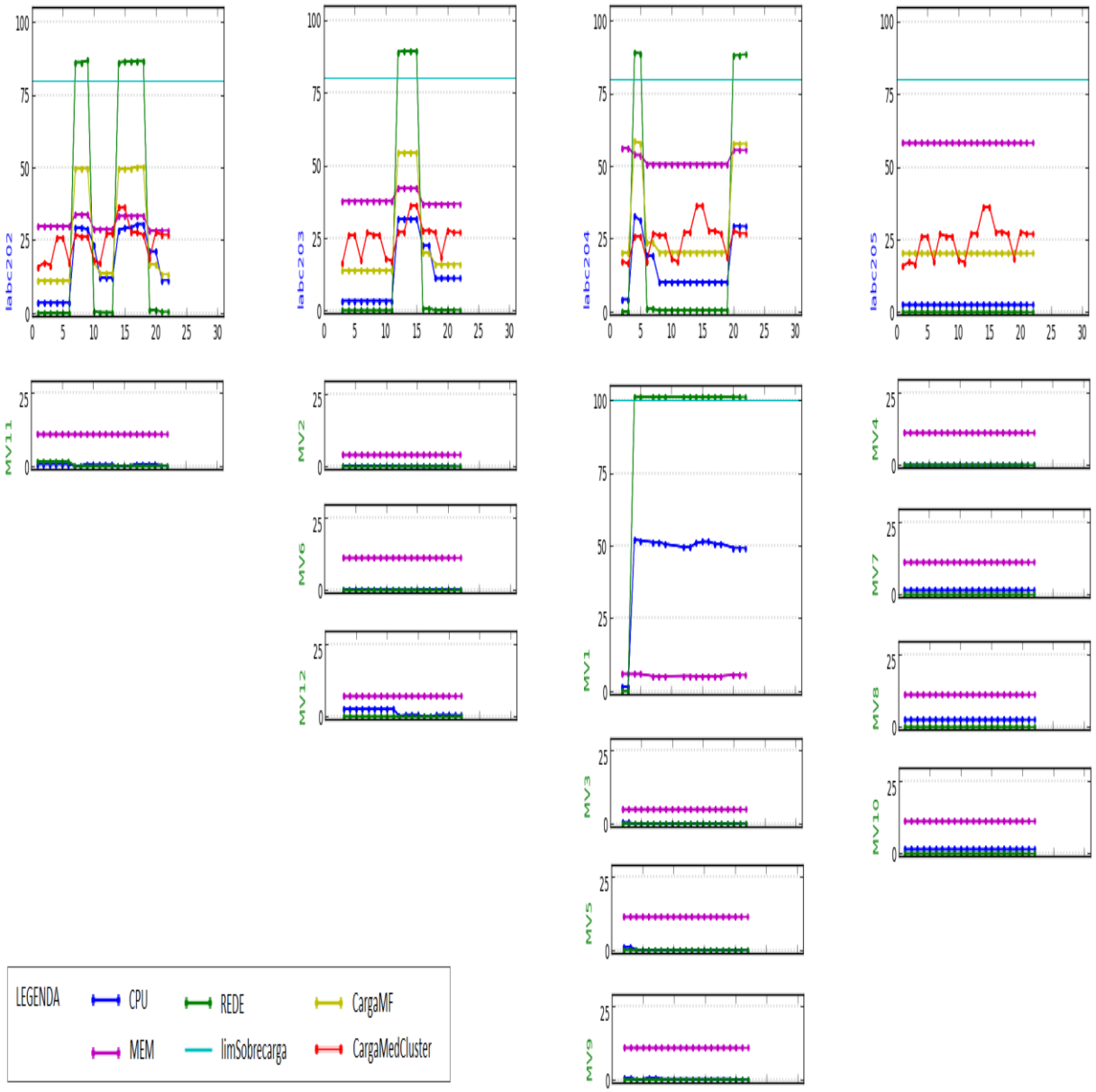

FIG. 5.32 Sobrecarga de Rede MV - Situação Final do Cluster

Este experimento mostrou que, quando uma MV com sobrecarga de Rede não é fixada na MF hospedeira onde foi detecta a sobrecarga, põe em risco a entrega de serviço de todo o cluster, mesmo quanto a MV mais sobrecarregada é a primeira a ser migrada. Isto ocorre, pois, ao ser migrada para outra MF, a MV sobrecarregada compromete outras MVs, impedindo que a MF de destino forneça os recursos necessários à demanda dos serviços hospedados.

5.5.4 RESULTADO DO EXPERIMENTO 3

Este experimento comparou o comportamento da arquitetura Phoenix (item 1) com o comportamento descrito na estratégia proposta no item 2 e com estratétigas definidas no item 3 que decreve o comportamento proposto por outros dois sistemas (Sandpiper e VOLTAIC) quando submetidos a sobrecarga de rede da MV, conforme se segue:

1. Quando a MV sobrecarregada é fixada na MF hospedeira, migrando as MVs por ordem de menor custo (estratégia da arquitetura Phoenix);
2. Quando a MV sobrecarregada não é fixada na MF hospedeira, migrando as MVs por ordem de menor custo (comportamento inspirado no Sandpiper); e
3. Quando a MV sobrecarregada não é fixada na MF hospedeira, migrando as MVs por ordem de maior custo (comportamento inspirado no VOLTAIC).

Quanto a primeira estratégia, esta se mostrou eficaz para o tratamento das sobrecargas de rede dentro cenário proposto, pois conseguiu isolar a MV com sobrecarga na MF hospedeira, permitindo a migração das MVs não afetadas e impedindo a propagação dos efeitos da sobrecarga nas MFs dos cluster.

A segunda estratégia não isolou a MV com sobrecarga na MF hospedeira, permitindo a propagação dos efeitos desta sobrecarga nas demais MFs do cluster. Como a MV migrada foi a de menor custo, as demais MFs do cluster demoraram mais tempo para sentir os efeitos da sobrecarga gerada pela MV sobrecarregada. Ao buscar mais recursos em outra MV sobrecarrega comprometeu a MF seguinte. Como ocorreria no caso do e Sandpiper (WOOD et al., 2009).

A terceira estratégia também não evitou o comprometimento das demais MVs ao não isolar a MV com sobrecarga na MF hospedeira, permitindo a propagação dos efeitos desta sobrecarga nas MFs do cluster. Mesmo que a MV sobrecarregada fosse para outra MF que aceitasse comportamento anômalo descrito pelo VOLTAIC (CARVALHO, 2012), as MVs que compartilham esta MF também sofreriam os efeitos dessa sobrecarga.

Em situações extremas, como a proposta pelo experimento 3, a arquitetura Phoenix poderia fazer a desativação da MV sobrecarregada até a migração das MVs que estivessem compartilhando a mesma MF, evitando o comprometimento do ANS dessas MVs. Isto poderia comprometer o ANS da MV afetada, mas ainda sim ficaria dentro do limite de *downtime* esperado para uma situação critica como esta.

5.6 ANÁLISE DOS RESULTADOS

O primeiro experimento visou analisar o comportamento da arquitetura em função da variação da CMC. Isto permitiu identificar a variação de CMC que seria utilizada para realização do balanceamento de carga da arquitetura compatível com a configuração do ambiente de testes proposto.

Cabe ressaltar que a parametrização aqui escolhida não esgota as possibilidades de configuração da arquitetura. Esta foi escolhida baseando-se em um cenário específico, definido para testes. Em outros cenários, possivelmente caberá outra definição da Faixa de Equilíbrio de acordo com as dimensões do cluster, número de migrações pretendidas e outras considerações do administrador.

O segundo experimento analisou o comportamento da arquitetura em face da sobrecarga da MF em relação à CPU e memória. A efetividade do algoritmo para tratamento de sobrecarga da MF foi demonstrada por meio do experimento que avaliou o comportamento da arquitetura quando submetida a sobrecarga espontâneas e a *oversubscription*.

As considerações aqui feitas em relação ao recurso CPU também são válidas para os recursos memória e rede, sendo que para o recurso memória, considerando o ambiente proposto para testes, recomenda-se um limiar de

utilização menor (70%) de modo que a memória virtual em disco seja menos utilizada e não prejudique as migrações das MVs.

O terceiro experimento avaliou o comportamento da arquitetura quando submetida a sobrecarga de rede na MV. Este avaliou a sobrecarga na rede configurada para cada MV considerando a utilização de rede da MF. Isto foi feito porque o recurso rede é compartilhado por meio da utilização da placa de rede de MF, sendo que sua utilização mesmo que definida como um percentual para cada MV tem sua geração de carga provocada a partir do meio externo, não controlado pelo hipervisor. Deste modo, se uma MV tiver uma capacidade de rede configurada e ultrapassar este limite, a MF também tratará estas requisições não previstas.

No caso do recurso rede, a MV afetada pode ser acometida por um excesso de requisições legítimas, mas não previstas, como *flash crowd*, por exemplo. Em situações como estas, é preciso isolar o problema de modo a não comprometer o ANS das demais MVs, até que possa ser estabelecida uma solução definitiva para o problema.

Neste experimento foi realizada uma comparação entre três estratégias, e somente aquela proposta na arquitetura Phoenix foi recomendada.

De maneira geral, foram feitos testes que concluíram que a proposta implementada funciona adequadamente em um cenário restrito e real. Neste cenário, foram capturadas informações de uso reais tanto das MFs quanto das MVs às quais foram submetidas a tratativas necessárias para cada situação apresentada. Observou-se que a arquitetura apresentou um comportamento eficaz tanto no tratamento de sobrecargas momentâneas como no balanceamento de carga no ambiente virtualizado.

5.7 COMPARAÇÃO COM OUTRAS ARQUITETURAS

Em relação ao tratamento de sobrecargas em ambiente de computação em nuvem, a fim de posicionar a arquitetura Phoenix entre as arquiteturas Sandpiper (WOOD et al., 2009) e VOLTAIC (CARVALHO, 2012), descritas no Capítulo 3, foi elaborada a TAB. 5.11 que estabelece uma síntese das principais características das arquiteturas relacionadas.

TAB. 5.11 Síntese das Características das Arquiteturas Relacionadas

Características	Sandiper	VOLTAIC	Phoenix
Tratamento de sobrecarga de CPU e Memória (MF, MV)	Sim	Sim	Sim
Tipo de Abordagem	Caixa Preda Caixa Cinza	Caixa Preda	Caixa Preta
Tipo de Migração	Live Migration	Live Migration	Live Migration
Tratamento de sobrecarga de rede	Sim	Não	Sim
Isolamento da MV com sobrecarga de rede	Não	Não	Sim
Uso da Libvirt	Não	Sim	Sim
Balanceamento de Carga	Nao	Sim	Sim

A partir da comparação da descrição dos comportamentos das arquiteturas Sandpiper (WOOD et al., 2009) e VOLTAIC (CARVALHO, 2012) com a Phoenix, observa-se que, apesar destas realizarem o tratamento de sobrecarga de CPU e memória de MFs e MVs, ambas não possuem mecanismos para isolamento das MVs com sobrecarga de rede. Assim, isto é um problema importante e não tratado por essas abordagens.

A Phoenix por sua vez, não possui métodos de predição existentes nas demais arquiteturas, porém além das outras funcionalidades apresentadas na TAB. 5.11, realiza o tratamento de sobrecarga de MV e MF e utiliza duas estratégias (proativa e reativa) conjuntas para tratamento de sobrecargas momentâneas.

Conclui-se, portanto, que a arquitetura Phoenix possui funcionalidades que, em conjunto, não são encontradas nas arquiteturas referenciadas para detecção e tratamento de sobrecargas de redes de MVs em ambiente de computação em nuvem. Estas funcionalidades fornecem uma nova abordagem para um problema já conhecido e de difícil solução. A FIG. 5.33 representa uma simulação de parte do comportamento da arquitetura Phoenix, em face de uma sobrecarga de rede.

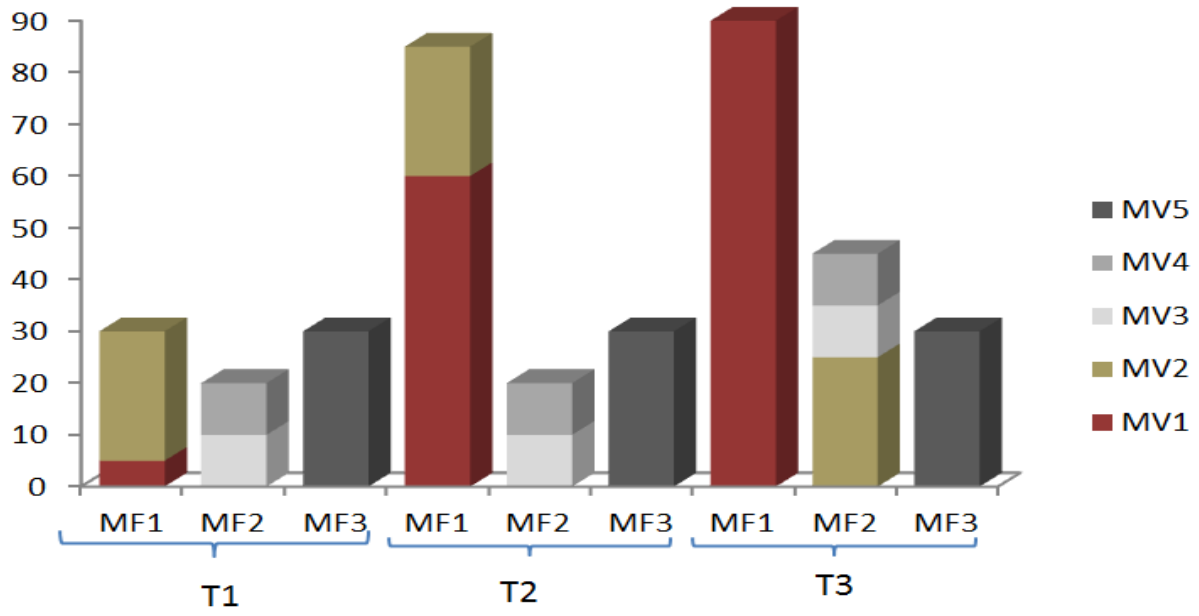

FIG. 5.33 Simulação de Comportamento da Phoenix em Relação à Sobrecarga de Rede

Como pode ser vista na FIG. 5.33, em T1, as MF1, MF2 e MF3 encontram-se funcionando normalmente. No tempo T2, a MV1 sofre uma sobrecarga de rede. No tempo T3, a MV1 é isolada na MF1, evitando a propagação da sobrecarga pelo cluster e comprometimento das demais MVs.

5.8 OPORTUNIDADES DE MELHORIA DA AQUITETURA PHOENIX

Além do potencial da Phoenix para tratar sobrecargas momentâneas, foram identificadas oportunidades de melhoria que caso sejam implementadas podem tornar a arquitetura mais completa.

A primeira oportunidade de melhoria diz respeito a módulo Analisador que originalmente foi concebido de forma única, tornando-se um ponto único de controle e em caso de falha, certamente comprometerá o funcionamento da arquitetura. Talvez a sua duplicação associada a uma política de eleição de analisadores para o controle dos nós do cluster possa reduzir significativamente o risco existente.

A segunda oportunidade seria um esquema de punição para MVs que ultrapassassem o limite de sobrecarga da MF no que diz respeito ao recurso

rede, deste modo poderia ser determinada a paralização da MV afetada até a migração das MVs que estejam compartilhando a mesma MF de modo a não comprometer os ANSs destas.

Outra oportunidade de melhoria seria a possibilidade de considerar o apoio à decisão multicritério para melhor justificar a seleção automática das MFs que recebem as MVs selecionadas para migração.

Atualmente parte da seleção automática é baseada na prioridade estabelecida pelo administrador por meio da atribuição de pesos para cada um dos recursos (CPU, memória, rede) que influenciam na decisão e isto é feito de acordo com sua experiência e observação dos arquivos de logs gerados pela arquitetura.

Uma outra oportunidade de melhoria, seria o implemento de algoritmos de inteligência artificial incluindo aprendizado de máquina de modo que pudessem ser feitos ajustes aumáticos de faixa de equilíbrio ou mesmo limiares de sobrecarga, dinamizando ainda mais o comportamento a arquitetura.

6 CONCLUSÃO

Como pode ser visto ao longo deste trabalho, as nuvens de computadores podem ser consideras como um conjunto de recursos virtualizados organizado para prestação de serviços sob demanda. Neste ambiente virtualizado, provido por mecanismos elásticos, podem surgir ameaças como: períodos de indisponibilidade e quedas de desempenho que podem ser prejudiciais aos serviços hospedados, comprometendo seriamente a credibilidade do provedor.

Mecanismos que automatizem processos de gerência e aumentem a eficiência dos provedores de serviços em nuvem são necessários para que os provedores consigam suportar os ANSs, minimizando eventuais prejuízos causados por sobrecargas ao funcionamento deste ambiente.

A gerência eficiente destes recursos computacionais que suportam os serviços pode trazer inúmeros benefícios, sendo o principal a manutenção dos serviços oferecidos dentro dos níveis acordados. A administração destes recursos representa um problema complexo que necessita de soluções viáveis.

Neste cenário, surge a necessidade de estratégias que visem o atendimento dos ANSs de modo a suportar eventuais e momentâneas sobrecargas visando prover elasticidade ao ambiente.

Este trabalho propôs a combinação de estratégias proativas e reativas para proporcionar uma melhor elasticidade nos ambientes de computação em nuvem de modo a suportar sobrecargas momentâneas.

Implementou-se a arquitetura computacional Phoenix utilizando as técnicas propostas, a qual foi capaz de viabilizar o tratamento de sobrecargas momentâneas neste ambiente, mostrando resultados favoráveis, principalmente no tratamento de sobrecargas de rede de MVs.

Foram realizados três conjuntos de experimentos que puderam comprovar a eficácia da arquitetura proposta. Foi possível ainda verificar que mesmo utilizando um cenário específico e real, é possível atingir os objetivos propostos ao suportar os efeitos dessas sobrecargas no ambiente virtualizado utilizado pela computação em nuvem.

Os experimentos demonstraram que a arquitetura Phoenix consegue identificar e tratar eventos de sobrecarga da MF que afetam os recursos (CPU,

memória e rede), tendo como referência os ANSs dos serviços de infraestrutura (IaaS) hospedados.

Fazendo uma análise comparativa, pode-se destacar que, em relação ao Sandpiper a Phoenix, possui a vantagem de detectar e tratar sobrecarga no uso da rede das MVs, isolando a MV sobrecarregada na origem, e de poder ser utilizada por vários hipervisores, não se limitando ao hipervisor Xen. Porém, possui a desvantagem de não possuir mecanismos de predição para sobrecargas.

Já em relação ao VOLTAIC, a Phoenix possui a vantagem de realizar o tratamento de sobrecargas de rede das MVs, isolando a MV sobrecarregada na MF hospedeira. O VOLTAIC migra a MV sobrecarregada para uma MF que abriga outras MVs com comportamento anômalo. Isto pode acarretar o comprometimento do ANS das MVs alocadas nesta MF.

Ainda em relação ao comparativo com o VOLTAIC, a Phoenix não utiliza um mecanismo que permita o desenvolvimento de controladores flexíveis para auxiliar a tomada de decisões de acordo com regras qualitativas, definidas pelos gerentes de nuvem. Na Phoenix, este mecanismo está restrito a atribuição de pesos para cada recurso avaliado no momento em que a arquitetura é parametrizada. Estes são utilizados na escolha da MF que receberá a MV escolhida para migração.

Cabe destacar que o processamento de clusters em nuvens computacionais exigem muitos outros recursos em termos de hardware e software, como sistemas de gerenciamento de recursos em nuvens, redes de alta velocidade e sistemas de armazenamento em disco. Por motivo de limitação dos recursos reais disponíveis, este trabalho utilizou um número reduzido de MFs, MVs, equipamentos de rede e armazenamento de modo a demonstrar a efetividade da proposta.

Adicionalmente, mesmo considerando que a arquitetura Phoenix tenha sido testada em um cenário restrito, ela pode ser adaptada para funcionar em diferentes cenários.

No cenário proposto por esta pesquisa é possível destacar as seguintes contribuições:

- Estudo comparativo das abordagens descritas nos trabalhos relacionados com foco no gerenciamento e tratamento de sobrecargas em ambiente de computação em nuvem;
- Estudo das estratégias e abordagens associadas à elasticidade em ambiente de computação em nuvem;
- Proposta de uma nova abordagem para classificação de algoritmo de balanceamento de carga em ambiente de computação em nuvem;
- Proposta de uma nova abordagem, baseada em mecanismo proativo e reativo, para tratamento de sobrecargas em ambiente de computação em nuvem;
- Desenvolvimento de algoritmos que viabilizaram a implementação da arquitetura proposta;
- Desenvolvimento de uma ferramenta capaz de materializar os objetivos da abordagem proposta;
- Criação de um ambiente real para pesquisa e continuados estudos; e
- Realização de experimentos que visaram aplicação prática das ideias propostas, demonstrando resultados significativos.

Durante o desenvolvimento desse trabalho, foram identificadas possibilidades de novos estudos a serem exploradas em trabalhos futuros. São elas:

- O uso de análise multicritérios para melhor qualificar/priorizar os recursos CPU, memória e rede analisados pela arquitetura;
- Estabelecimento da Faixa de Equilíbrio do cluster, considerando a CMC, tamanho do cluster e da política de administração;
- A aplicação desta abordagem em outro domínio mais amplo envolvendo mais de um cluster e múltiplos analisadores e uma política de eleição dos gerentes por datacenter ou região; e
- Ampliação da arquitetura para um ambiente em nuvem envolvendo escalabilidade horizontal.

7 REFERÊNCIAS BIBLIOGRÁFICAS

ADAMS, Keith; AGESEN, Ole. A comparison of software and hardware techniques for x86 virtualization.**ACM Sigplan Notices**, v. 41, n. 11, p. 2-13, 2006. Disponível em:< http://dl.acm.org/citation.cfm?id=1168860>. Acesso em:05 out 2013.

AGRAWAL, Divyakant et al. Database scalability, elasticity, and autonomy in the cloud. In:**Database Systems for Advanced Applications**. Springer Berlin Heidelberg, 2011. p. 2-15. e. Disponível em: <http://link.springer.com/chapter/10.1007/978-3-642-20149-3_2#page-1>. Acesso em: 9 set. 2014

AMD. **AMD Virtualization.** Disponível em: <http://www.amd.com/br/products/technologies/virtualization/Pages/virtualization.aspx>. Acesso em: 03 jun 2013.

AMOS WATERLAND. **Stress**. Workload generator and stress tester for Unix-like systems.Disponível em: < http://people.seas.harvard.edu/~apw/stress/ />. Acessado em: 29/05/2013.

ALAKEEL, Ali M. A guide to dynamic load balancing in distributed computer systems**. International Journal of Computer Science and Information Security**, v. 10, n. 6, p. 153–160, 2010. Disponível em: < http://paper.ijcsns.org/07_book/201006/20100619.pdf>. Acesso em 05 out 2013.

AMAZON INC. **Amazon web services Ec2.** Disponível em: <http://aws.amazon.com/pt/ec2>. Acesso em: 31 mar. 2013.

ANWER, Muhammad Bilal et al. Network I/O fairness in virtual machines. In: **Proceedings of the second ACM SIGCOMM workshop on Virtualized infrastructure systems and architectures**. ACM, 2010. p. 73-80. Disponível em:< http://dl.acm.org/citation.cfm?id=1851412>. Acesso em: 31 mar. 2013.

BARHAM, Paul et al. Xen and the art of virtualization. **ACM SIGOPS Operating Systems Review**, v. 37, n. 5, p. 164-177, 2003. Disponível em:< http://dl.acm.org/citation.cfm?id=945462f>. Acesso em: 31 mar. 2013.

BASET, Salman A.; WANG, Long; TANG, Chunqiang. Towards an understanding of oversubscription in cloud. In: **Proceedings of the 2nd USENIX conference on Hot Topics in Management of Internet, Cloud, and Enterprise Networks and Services**. USENIX Association, 2012. p. 7-7. Disponível em: <https://www.usenix.org/sites/default/files/conference/protected-files/oversubscribe-baset.pdf>. Acesso em: 1 fev. 2014.

BHADANI, A.; CHAUDHARY, S. Performance Evaluation of web servers using Central load Balancing Policy over Virtual Machines on Cloud. In: **Proceedings of the Third Annual ACM Bangalore Conference**. ACM, 2010. p. 16. Disponível em: < http://dl.acm.org/citation.cfm?id=1754304>. Acesso em: 1 fev. 2014.

BRISCOE, Gerard; MARINOS, Alexandros. Digital ecosystems in the clouds: towards community cloud computing. In: **Digital Ecosystems and Technologies, 2009. DEST'09. 3rd IEEE International Conference on**. IEEE, 2009. p. 103-108. Disponível em: < http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5276725&tag=1>. Acesso em: 1 fev. 2014.

BUGNION, Edouard et al. Bringing virtualization to the x86 architecture with the original vmware workstation. **ACM Transactions on Computer Systems (TOCS)**, v. 30, n. 4, p. 12, 2012. Disponível em: < http://dl.acm.org/citation.cfm?id=2382554>. Acesso em: 01 fev. 2014.

BUYYA, Rajkumar et al. Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility. **Future Generation computer systems**, v. 25, n. 6, p. 599-616, 2009. Disponível em: < http://www.sciencedirect.com/science/article/pii/S0167739X08001957>. Acesso em: 01 mar. 2013

CALHEIROS, R. N. et al. The Aneka platform and QoS-driven resource provisioning for elastic applications on hybrid Clouds. **Future Generation Computer Systems**, v. 28, n. 6, p. 861–870, jun. 2012. Disponível em: < http://www.sciencedirect.com/science/article/pii/S0167739X11001397>. Acesso em: 01 mar. 2013

CARISSIMI, Alexandre. Virtualização: da teoria a soluções. **Minicursos do Simpósio Brasileiro de Redes de Computadores–SBRC**, v. 2008, p. 173-207, 2008..Disponível em: <http://www.jvasconcellos.com.br/unijorge/wp-content/uploads/2012/01/cap4-v2.pdf>. Acesso em: 29 jul. 2013

CARVALHO, Hugo Eiji Tibana. **VOLTAIC: Um Sistema de Gerência Automatizada de Recursos de Nós Virtuais para Computaçao em Nuvens**. 2012. Dissertação de Mestrado. Universidade Federal do Rio de Janeiro. Disponível em: < http://www.pee.ufrj.br/teses/textocompleto/2012121401.pdf>. Acesso em: 29 jan. 2014.

CHANDRA, Abhishek; SHENOY, Prashant. Effectiveness of dynamic resource allocation for handling internet flash crowds. **TR03-37, Department of Computer Science, University of Massachusetts, USA**, 2003. Disponível em: < http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.124.2431&rep=rep1&type=pdf>. Acesso em: 21 jan. 2013.

FITÓ, Josep Oriol; GOIRI, Ínigo; GUITART, Jordi. SLA-driven elastic cloud hosting provider. In: **Parallel, Distributed and Network-Based Processing (PDP), 2010 18th Euromicro International Conference on, 2010**. IEEE. p. 111-118. Disponível em: < http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5452504&tag=1 f>. Acesso em: 23 jul. 2013.

COUTINHO, E.; SOUSA, F. COUTINHO. Elasticidade em computaçao na nuvem: Uma abordagem sistemática. **XXXI Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuıdos (SBRC 2013)-Minicursos**, 2013. Disponível em: <http://sbrc2013.unb.br/files/anais/minicursos/minicurso-5.pdf>. Acesso em: 23 jan 2014

DAWOUD, Wesam; TAKOUNA, Ibrahim; MEINEL, Christoph. Elastic vm for cloud resources provisioning optimization. In: **Advances in Computing and Communications**. Springer Berlin Heidelberg, 2011. p. 431-445. Disponível em: <http://www.springerlink.com/index/K75M2705443R2402.pdf>. Acesso em: 15 set. 2013.

DILLON, Tharam; WU, Chen; CHANG, Elizabeth. Cloud computing: issues and challenges. In: **Advanced Information Networking and Applications (AINA), 2010 24th IEEE International Conference on**. IEEE, 2010. p. 27-33. Disponível em: < http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5474674>. Acesso em: 15 mar. 2013.

FARAH, P.; MURTA, C. Efeitos de sobrecargas transientes em servidores web. In: 25° Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos **(** SBRC), 2007. Belém. **Anais..**. SBRC, 2007 p. 455–468. Disponível em: < http://ce-resd.facom.ufms.br/sbrc/2007/032.pdf>. Acesso em: 15 mar. 2013.

FERNANDES, Almir Dominicini. **Avaliação experimental de técnicas de virtualização através de balanceamento de carga em clusters de computadores**. 2010. Dissertação de Mestrado. Universidade Federal do Rio de Janeiro. Disponível em: < http://objdig.ufrj.br/60/teses/coppe_m/AlmirDominiciniFernandes.pdf>. Acesso em: 15 mar. 2013.

FERREIRA, P. H. S.; MARTINS, Y. C. **Balanceamento de Carga de Máquinas Virtuais em Ambiente Cloud Computing.** 2013. Monografia. Instituto Militar de Engenharia.. Rio de Janeiro - RJ.

FRANKE, H. **Uma Abordagem de acordo de nível de serviço para computação em nuvem**. 2010. Dissertação de Mestrado. Universidade Federal de Santa

Catarina. Santa Catarina, PN. Disponível em: < https://repositorio.ufsc.br/handle/123456789/94>. Acesso em: 15 mar. 2013.

GALANTE, Guilherme; BONA, Luis Carlos E. de. A survey on cloud computing elasticity. In: **Proceedings of the 2012 IEEE/ACM Fifth International Conference on Utility and Cloud Computing**. IEEE Computer Society, 2012. p. 263-270. Disponível em: < http://dl.acm.org/citation.cfm?id=2415736>. Acesso em: 15 mar. 2013.

GMACH, Daniel et al. Resource pool management: reactive versus proactive or let's be friends. **Computer Networks**, v. 53, n. 17, p. 2905-2922, 2009. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1389128609002655>. Acesso em: 16 mar. 2013.

GONG, Zhenhuan; GU, Xiaohui; WILKES, John. Press: predictive elastic resource scaling for cloud systems. In: **Network and Service Management (CNSM), 2010 International Conference on**. IEEE, 2010. p. 9-16. Disponível em: < http://dl.acm.org/citation.cfm?id=2415736>. Acesso em: 15 mar. 2013.

GOOGLE. **Psutil**: A cross-platform process and system utilities module for Python. Disponível em: "https://code.google.com/p/psutil/wiki/Documentation". Acesso em: 20 Ago 2013.

HERBST, N.; KOUNEV, S.; REUSSNER, R. Elasticity in cloud computing: what it is, and what it is not.Proceedings of the 10th International Conference on Autonomic Computing(ICAC 2013). **Anais**...San Jose, CA: 2013. Disponível em: <http://sdqweb.ipd.kit.edu/publications/pdfs/HeKoRe2013-ICAC-Elasticity.pdf>. Acesso em: 29 jul. 2014.

INTEL. **Intel® Virtualization Technology (Intel® VT**). Disponível em: <http://www.intel.com/technology/virtualization/technology.htm>. Acesso em: 03 jun 2013.

IPERF-FR. **IPERF**.:The TCP/UDP Bandwidth Measurement Tool. Disponível em < https://iperf.fr//>. Acesso em: 20 Ago 2013.

KANSAL, N. J.; CHANA, I. Existing load balancing techniques in cloud computing : a systematic review. **Journal of Information Systems and Communication,** v. 3, n. 1, p. 87–91, 2012. Disponível em: < http://www.chinacloud.cn/upload/2012-07/12071500212858.pdf>. Acesso em: 22 jul. 2013.

KHIYAITA, A. et al. Load balancing cloud computing: state of art. In: **Network Security and Systems (JNS2), 2012 National Days of**. IEEE, 2012. p. 106-109. Disponível em: < http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6249253&tag=1>. Acesso em: 22 jul. 2013.

KIVITY, A. et al. KVM: the linux virtual machine monitor**. Proceedings of the 2007 Linux Symposium**, p. 225–230, 2007. Disponível em: < https://ols.fedoraproject.org/OLS/Reprints-2007/OLS2007-Proceedings-V1.pdf#page=225>. Acesso em: 21 jan. 2013.

KNAUTH, Thomas; FETZER, Christof. Scaling non-elastic applications using virtual machines. In: **Cloud Computing (CLOUD), 2011 IEEE International Conference on**. IEEE, 2011. p. 468-475. Disponível em: < http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6008744&tag=1>. Acesso em: 22 jan. 2013.

MAGALHÃES, Deborah V.; SOARES, José Marques; GOMES, Danielo G. Análise do Impacto de Migração de Máquinas Virtuais em Ambiente Computacional Virtualizado. **Anais do XXIX SBRC**, 2011. Disponível em: < http://www.researchgate.net/profile/Danielo_Gomes/publication/234083570_Anlise_do_Impacto_de_Migrao_de_Mquinas_Virtuais_em_Ambiente_Computacional_Virtualizado_*/links/09e4150ef8e22eff2b000000.pdf >. Acesso em: 21 jan. 2013.

MARSHALL, Paul; KEAHEY, Kate; FREEMAN, Tim. Elastic site: Using clouds to elastically extend site resources. In: **Proceedings of the 2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing**. IEEE Computer Society, 2010. p. 43-52. Disponível em: < http://dl.acm.org/citation.cfm?id=1845214 >. Acesso em: 10 jan. 2013.

MELL, Peter; GRANCE, Timothy. The nist definition of cloud computing, recommendations of the national institute of standards and technolog. **National Institute of Standards and Technology**, p. 800-145, 2011. Disponível em: < http://faculty.winthrop.edu/domanm/csci411/Handouts/NIST.pdf>. Acesso em: 21 jan. 2013.

MISHRA, Ratan; JAISWAL, Anant. Ant colony optimization: A solution of load balancing in cloud. **International Journal of Web & Semantic Technology (IJWesT)**, v. 3, n. 2, p. 33-50, 2012. Disponível em: < http://www.airccse.org/journal/ijwest/papers/3212ijwest03.pdf >. Acesso em: 21 jan. 2013.

MOHARANA, Shanti Swaroop; RAMESH, Rajadeepan D.; POWAR, Digamber. Analysis of load balancers in cloud computing. **International Journal of Computer Science and Engineering**, v. 2, n. 2, p. 101-108, 2013. Disponível em: < http://www.researchgate.net/profile/Shanti_Moharana/publication/236521813_Analysis_of_Load_Balancers_in_Cloud_Computing/links/00b7d517fd8f65d3d1000000.pdf >. Acesso em: 21 jan. 2014.

NUAIMI, Klaithem Al et al. A survey of load balancing in cloud computing: Challenges and algorithms. In: **Network Cloud Computing and Applications (NCCA), 2012 Second Symposium on**. IEEE, 2012. p. 137-142. Disponível em:

<http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6472470>. Acesso em: 9 set. 2014

PADHY, R. P.; PATRA, M. R.; SATAPATHY, S. C. Virtualization techniques & technologies: State-of-the-art. **Journal of Global Research in Computer Science**, v. 2, p. 29–43, 2011. Disponível em: < http://www.jgrcs.info/index.php/jgrcs/article/view/269>. Acesso em: 21 jan. 2013.

PAN, C.; ATAJANOV, M. Flash crowds alleviation via dynamic adaptive network. In**: Proceeding of Internet Conference 2004.** p. 21-28, 2004. Disponível em: < https://www.internetconference.org/ic2004/PDF/paper/pan-chenyu.pdf>. Acesso em: 21 maio. 2013.

Python Software Foundation. **PYTHON**. Python Programming Language.. 2013. Disponível em: “http://www.python.org/”. Acesso em: 25 Ago 2013.

REDHAT **KVM**. Kernel Virtual Machine. Versão 2.1.0-rc1. Disponível em: < http://wiki.qemu.org/Download >. Acesso em: 20 Ago 2013.

REDHAT **LIBVIRT**. Virtualization API. Disponível em: <http://libvirt.org>. Acesso em: 20 Ago 2013.

REWEHEL, E. M.; MOSTAFA, M.-S. M. **A Survey on Load Techniques in Cloud Computing**. Internacional Journal of Engineering Research & Technolgy (IJERT), v. 3, n. 2, p. 178–183, 2014. Disponível em: < https://ols.fedoraproject.org/OLS/Reprints-2007/OLS2007-Proceedings-V1.pdf#page=225>. Acesso em: 21 jan. 2013.

SHARMA, Upendra et al. A cost-aware elasticity provisioning system for the cloud. In: **Distributed Computing Systems (ICDCS), 2011 31st International Conference on**. IEEE, 2011. p. 559-570. Disponível em: < http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5961733>. Acesso em: 21 jan. 2013.

SHARMA, Viney; SRIVASTAVA, Gur Mauj Saran. Energy efficient architectural framework for virtual machine management in IaaS clouds. In: **Contemporary Computing (IC3), 2013 Sixth International Conference on**. IEEE, 2013. p. 369-374..Disponível em: < http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6612222&tag=1>. Acesso em : 15 jan 2014

SHEN, Zhiming et al. Cloudscale: elastic resource scaling for multi-tenant cloud systems. In: **Proceedings of the 2nd ACM Symposium on Cloud Computing**. ACM, 2011. p. 5.. Disponível em: <http://dl.acm.org/citation.cfm?id=2038921>. Acesso em: 21 jan. 2013.

SIDHU, Amandeep Kaur; KINGER, Supriya. Analysis of load balancing techniques in cloud computing. **International Journal of Computers & Technology**, v. 4, n. 2, p. 737-741, 2013. Disponível em: < https://s3-ap-southeast-

1.amazonaws.com/erbuc/files/6dd21f70-77b7-4cff-bfc8-c806939f66bb.pdf >. Acesso em: 21 jan. 2014.

SILBERSCHATZ, A.; GALVIN, P. B.; GAGNE, G. **Operating System Concepts** (7thEdition). John Wiley & Sons, Inc., NJ, USA, 2005

SMITH, James E.; NAIR, Ravi. The architecture of virtual machines. **Computer**, v. 38, n. 5, p. 32-38, 2005. Disponível em: < http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1430629 >. Acesso em: 21 jan. 2013.

SOBESLAVSKY, Petr et al. Elasticity in cloud computing. **Master's thesis, Joseph Fourier University, ENSIMAG, Grenoble, France**, 2011. Disponível em: < http://www.sobeslavsky.net/files/elasticity.pdf >. Acesso em: 21 jan. 2013.

SOUSA, Flávio RC; MOREIRA, Leonardo O.; MACHADO, Javam C. Computação em nuvem: Conceitos, tecnologias, aplicações e desafios. **II Escola Regional de Computação Ceará, Maranhão e Piauí (ERCEMAPI)**, p. 150-175, 2009. Disponível em: < http://www.lia.ufc.br/~flavio/papers/ercemapi2009.pdf >. Acesso em: 21 jan. 2013.

SWARNKAR, N.; SINGH, A. K.; SHANKAR, D. R. A Survey of Load Balancing Techniques in Cloud Computing. **International Journal of Engineering Reseacg & Technology(IJERT)**, v. 2, n. 8, p. 800–804, 2013. Disponível em:< http://www.ijert.org/view-pdf/4721/a-survey-of-load-balancing-techniques-in-cloud-computing>.Acesso em : 22 jan 2014.

TUNCAY, E. Towards virtualization: A competitive business continuity. **African Journal of Business Management**, v. 4, n. 10, p. 2164–2173, 2010. Disponível em:< http://www.academicjournals.org/article/article1380797316_Ercan.pdf >.Acesso em : 22 jan 2014.

UHLIG, R. et al. Intel virtualization technology. **Computer**, v. 38, n. 5, p. 48–56, 2005. Disponível em:< http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1430631 >.Acesso em : 22 jan 2014.

VECCHIOLA, Christian; PANDEY, Suraj; BUYYA, Rajkumar. High-performance cloud computing: A view of scientific applications. In: **Pervasive Systems, Algorithms, and Networks (ISPAN), 2009 10th International Symposium on**. IEEE, 2009. p. 4-16. Disponível em:< http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5381983>.Acesso em : 22 jan 2013

VICTOR, Jeff. Solaris™ Containers Technology Architecture Guide. **Sun Blueprint, Mai**, 2006. Disponível em:< http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.119.1587&rep=rep1&type=pdf>.Acesso em : 22 jan 2013

VIRTUALIZATION, AMD64. Secure virtual machine architecture reference manual. **AMD Publication**, n. 33047, 2005.. Disponível em:< http://www.mimuw.edu.pl/~vincent/lecture6/sources/amd-pacifica-specification.pdf>. Acesso em: 31 mar. 2013.

WEBER, Andreas et al. Towards a Resource Elasticity Benchmark for Cloud Environments. In: **2nd International Workshop on Hot Topics in Cloud Service Scalability (HotTopiCS 2014).** ACM, março de 2014. Disponível em: <https://sdqweb.ipd.kit.edu/publications/pdfs/WeHeGrKo2014-HotTopicsWS-ElaBench.pdf>. Acesso em: 4 set. 2014

WOOD, T. et al. Sandpiper: Black-box and gray-box resource management for virtual machines. **Computer Networks**, v. 53, n. 17, p. 2923–2938, dez. 2009. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1389128609002035f>. Acesso em: 4 set. 2013

XEN. **Xen project**. Disponível em: < http://wiki.xenproject.org/wiki/Category:Manual>. Acesso em: 20 Ago 2013.

ZHAO, Y. Z. Y.; HUANG, W. H. W. Adaptive Distributed Load Balancing Algorithm Based on Live Migration of Virtual Machines in Cloud. **2009 Fifth International Joint Conference on INC, IMS and IDC**, 2009. Disponível em:< http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5331732>. Acesso em: 05 set 2013.