Universidade do Estado do Rio de Janeiro
Centro de Tecnologia e Ciências
Faculdade de Engenharia

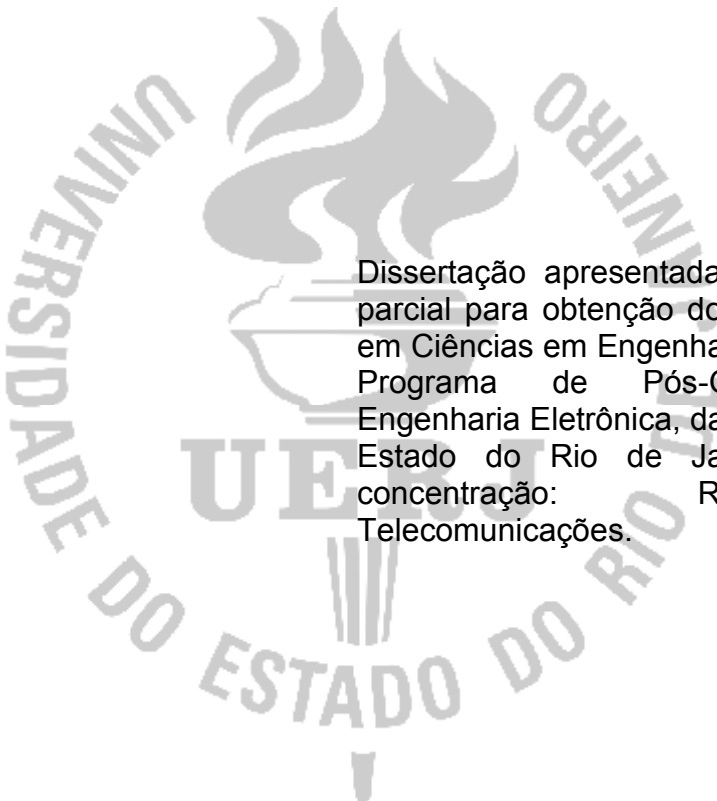
André Felipe de Almeida Monteiro

Utilização de redes neurais para gerência de servidores virtuais Web

Rio de Janeiro
2011

André Felipe de Almeida Monteiro

Utilização de redes neurais para gerência de servidores virtuais Web



Dissertação apresentada, como requisito parcial para obtenção do título de Mestre em Ciências em Engenharia Eletrônica, ao Programa de Pós-Graduação em Engenharia Eletrônica, da Universidade do Estado do Rio de Janeiro. Área de concentração: Redes de Telecomunicações.

Orientador: Prof. Dr. Alexandre Sztajnberg

Rio de Janeiro

2011

CATALOGAÇÃO NA FONTE
UERJ / REDE SIRIUS / BIBLIOTECA CTC/B

A994 Monteiro, André Felipe de Almeida.
Utilização de redes neurais para gerência de servidores virtuais Web / André Felipe de Almeida Monteiro. - 2011.
89 f. : il.

Orientador: Alexandre Sztajnberg.
Dissertação (Mestrado) – Universidade do Estado do Rio de Janeiro, Faculdade de Engenharia.

1. Redes de computação – Teses. 2. Virtualização – Teses. 3. Gerência de recursos – Teses. 4. Servidores *web* – Teses. 5. *Cluster*s de computadores – Teses. 6. Engenharia Eletrônica. I. Sztajnberg, Alexandre. II. Universidade do Estado do Rio de Janeiro. III. Título.

CDU 004.72.057.4

_____ _____

Assinatura Data

André Felipe de Almeida Monteiro

Utilização de redes neurais para gerência de servidores virtuais Web

Dissertação apresentada, como requisito parcial para obtenção do título de Mestre em Ciências em Engenharia Eletrônica, ao Programa de Pós-Graduação em Engenharia Eletrônica, da Universidade do Estado do Rio de Janeiro. Área de concentração: Redes de Telecomunicações.

Aprovada em _____

Banca examinadora:

_____

Prof. Dr. Alexandre Sztajnberg (Orientador)

Faculdade de Engenharia - UERJ

_____

Prof. Dr. Luiz Biondi Neto

Faculdade de Engenharia - UERJ

_____

Prof. Dr. Julius César Barreto Leite

Instituto de Computação - UFF

Rio de Janeiro

2011

AGRADECIMENTOS

Primeiramente, agradeço a minha família que me apoiou de forma integral durante toda esta jornada. Sem eles ao meu lado, esta etapa de minha vida não poderia ser finalizada.

A minha companheira de mestrado Leila Negris. Juntos nesta longa caminhada, o apoio mútuo foi fundamental nos momentos de dificuldade.

Aos meus alunos de Graduação em Ciência de Computação da UERJ, que a cada aula mostram que o processo de aprendizado é eterno e contínuo.

Principalmente ao meu orientador e professor Alexandre Sztajnberg. Suas palavras certeiras e conselhos sabiamente colocados foram fundamentais para a certeza de que eu não estava só nesta empreitada.

RESUMO

MONTEIRO, André Felipe de Almeida. *Utilização de redes neurais para gerência de servidores virtuais Web*. 90 f. Dissertação (Mestrado em Engenharia Eletrônica) - Programa de Pós-Graduação em Engenharia Eletrônica, Universidade do Estado do Rio de Janeiro, Rio de Janeiro, 2011.

Este trabalho propõe o uso de uma rede neural artificial, com utilização de mapas auto-organizáveis, para gerência de recursos de um *cluster* de servidores virtuais *web*. A função da rede neural consiste em classificar os estados de operação do *cluster* de acordo com três métricas: tempo de resposta, consumo de energia e carga de requisições. Assim, obtida a classificação corrente de operação, são realizadas intervenções no ambiente através da entrega ou retirada de recursos do *cluster*, com o objetivo de manter a qualidade de serviço da aplicação suportada pelos servidores virtuais dentro dos padrões pré-estabelecidos e promover um menor consumo de energia do ambiente. O estudo descreve a política de gerência de recursos implementada, além da arquitetura e configuração da rede neural utilizada. Também é apresentada uma avaliação de desempenho do sistema proposto, onde é indicado que a solução desenvolvida apresenta melhores resultados que a arquitetura original utilizada como base neste trabalho.

Palavras-chave: Virtualização. Redes Neurais Artificiais. Economia de Energia. Gerência de Recursos.

ABSTRACT

This work proposes using a neural network with self organizing maps, to build a configuration policy, which enables the management of a supporting infrastructure for *Web* applications using virtual machines. The neural network uses response time, power consumption and workload rate data collected from the cluster to classify the cluster´s operation states. In runtime, based on the classified current state a management architecture performs configuration operations adding or subtracting resources from the cluster. The overall goal is to ensure quality of service, while improving the energy saving compared to the original architecture in which the proposed architecture was based, acting more efficiently on physical servers (*host*s) or manipulating the virtual machines. The work includes a performance evaluation carried out over a system implemented based on the proposed architecture. This evaluation shows that the new configuration policy is more efficient than the original one, presenting better results for response time and energy saving.

Keywords: Virtualization. Neural Networks. Energy Saving. Resource Management. *Web* Servers.

LISTA DE FIGURAS

FIGURA 2 – CONSOLIDAÇÃO DE SERVIDORES ATRAVÉS DA VIRTUALIZAÇÃO 15
FIGURA 3 – ESTRUTURA DA CAMADA DE EMULAÇÃO 25
FIGURA 4 – CAMADA DE VIRTUALIZAÇÃO COMPLETA 26
FIGURA 5 – CAMADA DE PARAVIRTUALIZAÇÃO 26
FIGURA 6 – ARQUITETURA IMPLEMENTADA PELO XEN 28
FIGURA 7 – EXEMPLO DE UMA REDE NEURAL ARTIFICIAL 30
FIGURA 8 – (A) ATIVAÇÃO DOS NEURÔNIOS PARA O PADRÃO DE ENTRADA 1 33
FIGURA 8 – (B) ATIVAÇÃO DOS NEURÔNIOS PARA O PADRÃO DE ENTRADA 2 33
FIGURA 9 – SCRIPT PARA TREINAMENTO DE UMA REDE NEURAL NA FERRAMENTA SNNS 38
FIGURA 10 – ARQUITETURA IMPLEMENTADA 41
FIGURA 11 – INFRA-ESTRUTURA UTILIZADA 47
FIGURA 13 – TAXA DE REQUISIÇÕES SUBMETIDAS AO CLUSTER NO WORKLOAD EM RAMPA 48
FIGURA 14 – ARQUITETURA DA REDE NEURAL CONSTRUÍDA NESTE TRABALHO 49
FIGURA 16 – MAPA TOPOGRÁFICO GERADO APÓS A FASE DE TREINAMENTO DA REDE NEURAL 51
FIGURA 17 – COMPORTAMENTO DO WORKLOAD DA COPA DO MUNDO DE 1998 52
FIGURA 19 – MAPA TOPOGRÁFICO FINAL GERADO APÓS A FASE DE TREINAMENTO DA REDE NEURAL 54
FIGURA 20 – IMPACTOS DAS AÇÕES POSSÍVEIS NA REGIÃO 1 57
FIGURA 21 – IMPACTOS DAS AÇÕES POSSÍVEIS NA REGIÃO 2 PARA O CENÁRIO DE CARGA CRESCENTE 58
FIGURA 22 – IMPACTOS DAS AÇÕES POSSÍVEIS NA REGIÃO 2 PARA O CENÁRIO DE CARGA DECRESCENTE 59
FIGURA 23 – IMPACTOS DAS AÇÕES POSSÍVEIS NA REGIÃO 4 PARA O CENÁRIO DE CARGA DECRESCENTE 61
FIGURA 24 – IMPACTOS DAS AÇÕES POSSÍVEIS NA REGIÃO 4 PARA O CENÁRIO DE CARGA CRESCENTE 62
FIGURA 25 – IMPACTOS DAS AÇÕES POSSÍVEIS NA REGIÃO 5 63
FIGURA 26 – MÁQUINA DE ESTADOS DA NOVA POLÍTICA DE RECONFIGURAÇÃO 64
FIGURA 27 – EXEMPLO DO ARQUIVO MONTADO SUBMETIDO PARA PROCESSAMENTO DA REDE NEURAL 67
FIGURA 28 – TEMPO DE RESPOSTA PARA O WORKLOAD EM RAMPA 68
FIGURA 29 – POTÊNCIA CONSUMIDA PARA O WORKLOAD EM RAMPA 69
FIGURA 30 – TEMPO DE RESPOSTA PARA O WORKLOAD DA COPA DE 98 71
FIGURA 31 – CONSUMO DE ENERGIA PARA O WORKLOAD DA COPA DE 98 72
FIGURA 32 – COMPORTAMENTO DO WORKLOAD COM LOGS DE ACESSO À PÁGINA DA NASA 75
FIGURA 33 – TEMPO DE RESPOSTA PARA O WORKLOAD DA NASA 75
FIGURA 34 – CONSUMO DE ENERGIA PARA O WORKLOAD DA NASA 76
FIGURA 35 – CONSUMO DE CPU DOS SERVIDORES HOSPEDEIROS COM O O WORKLOAD DA COPA DE 98 78
FIGURA 36 – CONSUMO DE CPU DOS SERVIDORES HOSPEDEIROS COM O WORKLOAD DA NASA 78

LISTA DE TABELAS

TABELA 1 – FONTES ENERGÉTICAS BRASILEIRAS 14
TABELA 12 – CONFIGURAÇÃO DOS SERVIDORES FÍSICOS UTILIZADOS. 47
TABELA 15 – PARÂMETROS INICIAIS DEFINIDOS PARA A REDE NEURAL 50
TABELA 18 – PARÂMETROS FINAIS DEFINIDOS PARA A REDE NEURAL 53
TABELA 20 – VALORES PERCENTUAIS DE CADA MÉTRICA PARA AS REGIÕES CLASSIFICADAS 55

LISTA DE ABREVIATURAS

API	Application Program Interface
CPU	Central Processing Unit
DVFS	Dynamic Voltage and Frequency Scaling
HTML	Hyper-Text Markup Language
HTTP	Hyper-Text Transfer Protocol
IC	Intervalo de confiança
IP	Internet Protocol
MAC	Medium Access Control
MMV	Monitor de Máquina Virtual
NASA	National Aeronautics and Space Administration
NFS	Network File System
QoS	Quality of Service
RGB	Red Green Blue
RNA	Rede Neural Artificial
SMP	Symmetric Multi-Processing
SO	Sistema Operacional
SOM	Self Organizing Maps
TCP	Transmission Control Protocol
TI	Tecnologia da Informação
URL	Uniform Resource Locator
USB	Universal Serial Bus
VA	Volt-ampere
W	Watts
XML	Extensible Markup Language

SUMÁRIO

INTRODUÇÃO **13**
Organização do texto **17**

1 TRABALHOS RELACIONADOS **18**
1.1 Virtualização **18**
1.2 Redes neuaris **19**
1.3 Discussão das soluções apresentadas **21**

2 VIRTUALIZAÇÃO E REDES NEURAIS ARTIFICIAIS **23**
2.1 Características e evolução da virtualização **23**
2.2 Técnicas de virtualização **24**
2.2.1 Emulação 24
2.2.2 Virtualização completa 25
2.2.3 Paravirtualização 26
2.3 A ferramenta de virtualização Xen **27**
2.4 Redes neurais artificiais **29**
2.4.1 Treinamento de uma rede neural 30
2.4.2 Mapas auto-organizáveis 32
2.4.3 A ferramenta SNNS (*Stuttgart Neural Network Simulator)* 36

3 SISTEMA PROPOSTO **39**
3.1 Premissas e diretrizes da proposta **39**
3.2 Arquitetura e API originais **41**
3.3 Validação da rede neural proposta **46**
3.4 Nova arquitetura e política de reconfiguração **51**
3.4.1 Configuração da rede neural 51
3.4.2 Nova política de reconfiguração 55
3.4.2.1 Região 1 56
3.4.2.2 Região 2 57
3.4.2.3 Região 3 59
3.4.2.4 Região 4 60
3.4.2.5 Região 5 62
3.4.3 Máquina de estados da nova política de reconfiguração 63

4 AVALIAÇÃO DE DESEMPENHO 66
4.1 Configuração dos testes 66
4.2 Workload em rampa 68
4.3 Workload com logs da Copa do Mundo de 1998 71
4.4 Workload com logs de acesso da NASA 75
4.5 Considerações sobre consumo de CPU 77
5 CONCLUSÃO E TRABALHOS FUTUROS 80
REFERÊNCIAS 83
APÊNDICE A - Funções da API 86
APÊNDICE B - Ações efetivadas pela API em cada região 88

INTRODUÇÃO

A preocupação com os níveis globais elevados de consumo de energia é crescente. Nos principais países do mundo, atualmente as pessoas estão consumindo mais energia do que em qualquer outra época. Outro fator preocupante é que o aumento no consumo de energia vem seguido de uma utilização de fontes energéticas altamente poluentes. As principais fontes de energia dos países desenvolvidos, que são também os maiores consumidores de energia, são baseadas em combustíveis fósseis, com destaque para o petróleo. A utilização desses elementos para o processo de geração de energia é o principal fator causador do efeito estufa. Conforme apontado em [1], as emissões de gases poluentes causadores do efeito estufa aumentaram em 70% no período de 1970 a 2004.

Com base nesse cenário preocupante, as principais potências mundiais, com presença do Brasil inclusive, tentam adotar medidas que reduzam a taxa de emissão de gases poluentes. As reuniões de cúpula de Estado realizadas nas cidades de Kyoto em 2002 e Copenhagen em 2009 foram projetadas para tratarem exclusivamente deste tema, mas os avanços em relação aos compromissos de redução das emissões foram insuficientes, e o panorama continua requerendo muita atenção.

Tabela 1 – Fontes energéticas brasileiras

Tipo	Usinas	Capacidade (MW)	%
Hidroelétrica	802	78.016	69,03
Gás	121	11.844	10,48
Petróleo	800	5.553	4,91
Biomassa	331	5.563	4,92
Nuclear	2	2.007	1,78
Carvão Mineral	8	1.455	1,29
Eólica	33	414	0,37
Importação	-	8.170	7,23

O Brasil se destaca no cenário mundial por ter uma matriz energética limpa, baseada na utilização de hidroelétricas, que apesar de causarem um grande impacto ambiental para sua implantação, se caracterizam por uma geração energética não poluidora. Cabe ressaltar que o grande responsável pela emissão de gases poluentes em nosso país é o desmatamento através de queimadas, principalmente na região amazônica. Entretanto, na questão energética o Brasil

apresenta uma planta de geração dentre as menos poluentes do mundo. Segundo [2], a matriz energética do país encontra-se dividida conforme a Tabela 1, onde é possível verificar que as fontes mais poluentes de geração de energia, como petróleo e gás, não possuem presença significativa em território nacional.

Com a alta nos preços da energia elétrica observada nos últimos anos, a adoção de medidas que otimizem a utilização deste insumo é pauta das principais deliberações de muitas empresas. Dentre os maiores consumidores de energia no segmento de indústria/serviços estão as empresas de Tecnologia da Informação (TI) que disponibilizam *datacenters* para utilização própria e/ou de terceiros. Em [4], há projeção na qual indica que em 2015 o custo relacionado ao consumo de energia em *datacenters* será igual ao custo de aquisição/manutenção dos equipamentos de TI, com viés de alta para os anos seguintes.

Conforme descrito em [5], atualmente o consumo de energia de um *datacenter* está divido em: 40% para os equipamentos de TI e 60% para infra-estrutura do ambiente. O grande agente deste desequilíbrio é a refrigeração desses ambientes, pois a dissipação da energia elétrica consumida por um servidor é realizada através do calor. Assim, como o desempenho dos servidores está diretamente relacionado à temperatura controlada de trabalho do processador e dos principais dispositivos de *hardware*, o ambiente externo da máquina deve estar muito bem refrigerado para que o calor dissipado não impacte de forma negativa no desempenho da máquina.

Para tratar a questão de dissipação do calor gerado pelos servidores de forma eficiente, o ambiente físico de um *datacenter* é projetado especialmente para otimizar a refrigeração do mesmo. Além deste aspecto, a consolidação dos servidores em racks especiais que facilitam a troca de calor com o ambiente, e uma distribuição adequada desses racks dentre o espaço disponível são medidas que contribuem de forma efetiva para um consumo de energia em menor escola dentro dos *datacenters*. A principal métrica utilizada para avaliação da eficiência energética de um centro de processamento é a DCIE (*Data Center Infrastructure Efficiency*), que é descrita na Equação 1. Esta equação indica que quando o valor do DCIE estiver próximo do valor 1, grande parte da energia fornecida ao ambiente está sendo consumida pelos equipamentos de TI (servidores e equipamentos de rede). Todavia, caso o DCIE indique valores distantes de 1, isso significa que o *datacenter* não apresenta uma boa eficiência energética, pois grande parte da energia consumida está destinada à infra-estrutura do ambiente.

$$DCIE = \frac{\text{Energia entregue para equipamentos de TI}}{\text{Energia entregue ao } \textit{datacenter}} \quad (1)$$

Dentre as soluções de TI disponíveis para tratamento deste tema, a consolidação de servidores físicos em um quantitativo menor de equipamentos através das técnicas de virtualização se mostra a mais eficiente. O estudo realizado em [6] indica que implantar a virtualização em um *datacenter* já existente, sem modificar os parâmetros de energia e refrigeração do mesmo, causa uma redução imediata do DCIE. Ou seja, a utilização de virtualização sempre apresenta um menor consumo de energia se comparado ao ambiente original que não fazia uso deste procedimento. A explicação deste resultado é simples, pois conforme indicado na Figura 2, a utilização da virtualização possibilita uma menor quantidade de servidores físicos ativos, já que esse quantitativo menor de equipamentos hospedará diversos servidores virtuais que disponibilizam os mesmos serviços presentes na configuração original do *datacenter*.

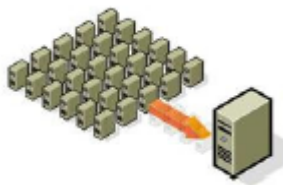


Figura 2 – Consolidação de servidores através da virtualização

Essa redução do número de servidores físicos é viabilizada também em virtude de grande parte dos equipamentos em operação em um *datacenter* serem super dimensionados, para que os mesmos estejam aptos a suportar um cenário de carga/ocupação referente ao limite projetado. Desta forma, na maior parte do tempo esses servidores apresentam recursos ociosos, que não podem ser disponibilizados em tempo real para outros equipamentos. Como na virtualização a disponibilização dos recursos de *hardware* pode ser realizada de forma dinâmica, cada servidor virtual é geralmente configurado para atender ao nível de carga/ocupação corrente, evitando assim a situação de recursos ociosos no ambiente, pois os mesmos podem ser utilizados por outro servidor virtual caso seja necessário.

Apesar da utilização de virtualização já possibilitar uma redução significativa do consumo de energia de um *datacenter*, é possível avançar nesta questão através de mecanismos de gerência dos servidores virtuais que sejam baseados em políticas relacionadas ao consumo energético. Esses mecanismos devem intervir no funcionamento dos servidores

virtuais para garantir que os serviços providos pelos mesmos sejam mantidos dentro dos valores especificados e promover um menor consumo de energia do ambiente computacional.

Este trabalho se propõe a desenvolver uma política de gerenciamento de servidores virtuais baseado em mecanismos inteligentes de gerência, que sejam capazes de identificar o cenário corrente de funcionamento do ambiente e intervir para otimização do consumo de energia do mesmo, respeitando os níveis de qualidade de serviço especificados. Para implementação desta política é utilizado um ambiente simplificado de *datacenter*, constituído de um aglomerado de servidores virtuais *Web* (ou *cluster* de servidores virtuais *web*), que são suportados por um conjunto de servidores físicos hospedeiros, e oferecem serviços específicos para um conjunto simulado de clientes. A arquitetura do *cluster* possui elementos distintos, com destaque para um módulo Controlador que deve garantir qualidade de serviço e buscar um modo de operação com maior economia de energia. Essas premissas são alcançadas através de intervenções nos servidores físicos, por meio de uma API desenvolvida baseada na Xen API.

A política originalmente construída em conjunto com a arquitetura acima mencionada foi desenvolvida por [7] e funciona basicamente em função da qualidade de serviço medida. Por meio de medições cíclicas do tempo de resposta dos serviços fornecidos pelo *cluster* é possível aumentar os recursos computacionais, caso o tempo de resposta esteja acima de um valor estipulado, ou caso o valor medido esteja abaixo daquele projetado, opta-se pela diminuição dos recursos disponíveis para prover um menor consumo de energia do conjunto de hospedeiros físicos. Para tornar a gerência do ambiente mais eficiente e melhorar os resultados obtidos pela arquitetura original, ou seja, obter tempo de resposta e consumo de energia com valores menores, este trabalho propõe a utilização de uma nova política de gerenciamento de recursos do *cluster*, onde, além do tempo de resposta, serão utilizados outros dois indicadores: (i) consumo de energia dos servidores físicos hospedeiros e (ii) taxa de requisições submetidas ao conjunto de servidores virtuais *Web*.

Com esta nova abordagem na política de reconfiguração, o processo de tomada de decisão utilizará a técnica de classificação de padrões (*clustering*) oferecida pelos mapas auto-oganizáveis (SOM - *Self Organizing Maps*, ou redes de Kohonen), que é um tipo de rede neural artificial. Na rede neural implantada as variáveis tempo de resposta, consumo de energia e carga submetida são combinadas em uma análise única que tem por objetivo indicar as melhores intervenções no ambiente. Com essa metodologia é possível classificar o estado

corrente do *cluster* com base nos três parâmetros mencionados, e a partir desta classificação é efetuada a tomada de decisão da nova política de reconfiguração proposta.

Para avaliação da presente proposta é realizada a comparação dos resultados obtidos através da solução implementada neste estudo com aqueles verificados durante a utilização da arquitetura original. Este procedimento é efetuado com a utilização, em ambas as propostas, do mesmo padrão de carga de requisições submetidos ao conjunto de servidores virtuais, de forma a viabilizar a comparação dos valores médios observados de consumo de energia e tempo de resposta entre os dois trabalhos. Desta forma, observa-se através dos experimentos realizados que a solução apresentada neste estudo provê uma redução média de até 26,4% do consumo de energia e 29,5% do tempo de resposta descrito na solução original.

Organização do texto

O presente trabalho está estruturado da seguinte forma: no Capítulo 1 são indicados os principais trabalhos relacionados com a gerência do consumo de energia de *cluster* de servidores *Web*, onde serão analisados estudos que se utilizam de ferramentas de virtualização e redes neurais, com o objetivo de prover a gerência e controle de recursos computacionais e melhoria do consumo de energia do ambiente de processamento. No Capítulo 2 são apresentados conceitos básicos relacionados às principais técnicas de virtualização e como as mesmas implementam seus serviços, além de apresentar os principais modelos de redes neurais artificiais. No Capítulo 3 é apresentada a arquitetura original utilizada e a proposta de solução desenvolvida neste trabalho, indicando as alterações efetuadas no módulo Controlador para implantação da nova política de reconfiguração e sua interface com a rede neural. No Capítulo 4 são apresentados os testes realizados para validação da proposta de solução, onde são comparados os resultados obtidos com o sistema desenvolvido neste trabalho com aqueles observados na arquitetura original. Finalmente, no capítulo 5 são discutidas as considerações finais e as conclusões deste estudo, além dos apontamentos para trabalhos futuros.

CAPÍTULO 1 TRABALHOS RELACIONADOS

Neste capítulo são apresentados os trabalhos relacionados aos temas de economia de energia em ambientes de processamento de dados e utilização de redes neurais como ferramenta de controle de recursos computacionais. As soluções mencionadas neste capítulo para tratamento da questão de economia de energia englobam tanto a utilização de virtualização de servidores para consolidação de equipamentos físicos quanto ambientes sem uso de virtualização. O uso de redes neurais para auxílio de monitoramento e análise de ambientes de processamento é realizado de formas distintas, tendo como objetivo comum melhorar e automatizar mecanismos de gerência e controle de um ambiente computacional.

1.1 Virtualização

Diferentes abordagens sobre o uso de virtualização e controle do uso de recursos computacionais, visando direta ou indiretamente às questões da qualidade de serviço e economia de energia estão presentes em outros trabalhos. É proposto em [8] uma arquitetura de gerência que provê balanceamento de carga utilizando migração de máquinas virtuais (VMs) em um conjunto de *host*s. Quando a utilização de CPU de um dos *host*s está acima de um limite previamente definido, o sistema migra as VMs para outros *host*s com mais recursos disponíveis. Desta forma, a proposta utiliza o uso de recursos dos *host*s físicos no processo de tomada de decisão para reconfiguração do ambiente. No estudo apresentado em [9] é utilizado um controlador *lookahead* que, baseado num conjunto de variáveis, tais como, quantidade de conexões e tempo de resposta médio, entre outras, ajusta o ambiente definindo a quantidade de *host*s que devem ser ativados e a quantidade de recursos que deve ser alocada para cada VM.

No trabalho apresentado por [10] é realizada a adaptação dinâmica de sistemas virtualizados num ambiente composto por múltiplos domínios. Cada domínio é formado por um *host* e suas VMs. A estratégia adotada foi a de alocar recursos paras as VMs ajustando-se o peso de CPU que o virtualizador atribui para cada uma delas de acordo com a demanda, e ajustando a quantidade de memória que a VM necessita. Caso em um domínio não seja possível alocar mais recursos para as VMs, o sistema pode buscar outro domínio que possua recursos disponíveis para realizar a migração de VMs. O sistema desenvolvido em [11] também se utiliza da carga de requisições submetida a um conjunto de servidores físicos, sem

uso de virtualização, para definição de uma política preditiva que tem por finalidade proceder a redução do consumo de energia do ambiente. A previsão da taxa de requisições submetidas ao ambiente de processamento é realizada com base em uma série histórica, e as intervenções nos servidores para diminuição do consumo de energia em cenários de baixa carga são efetuadas através do mecanismo DVFS, que altera a freqüência de operação do processador. Desta forma, o mecanismo de adaptação é baseado na utilização de CPU e de memória das VMs.

O trabalho descrito em [12] propõe uma arquitetura para adaptação dinâmica em servidores *Web* considerando os aspectos de economia de energia e qualidade de serviço. Na proposta, um aglomerado de servidores *Web* formados por servidores físicos, sem utilização de mecanismos de virtualização, recebe conexões através de um balanceador de carga (*frontend*). Os servidores físicos têm suas freqüências de CPU ajustadas dinamicamente e servidores ociosos são desligados, tudo de acordo com o tempo de resposta da aplicação. Em [13] é proposto um sistema que altera dinamicamente as configurações do SMP *Credit Scheduler*, que é o escalonador padrão do Xen que permite o balanceamento de processadores virtuais junto aos processadores reais do hospedeiro físico. O sistema implementado realoca recursos destinados a máquinas virtuais que não estejam utilizando todo o processamento disponibilizado originalmente, direcionando os recursos ociosos para máquinas virtuais que necessitem de uma fatia maior de processamento.

1.2 Redes neurais

A utilização de redes neurais artificiais com mapas auto-organizáveis permaneceu durante muito tempo limitada a processos de modelagem e reconhecimento de imagens e sons. Como a principal funcionalidade desta técnica de rede neural é classificar os valores de entrada de acordo com os padrões estabelecidos na fase de treinamento, a associação de imagens e sons perante outros elementos semelhantes pode ser implementada sem muito esforço pelos mapas auto-organizáveis. O trabalho apresentado em [14] descreve a construção de uma rede neural artificial com a finalidade específica de proceder o reconhecimento de imagens. Para que a ferramenta desenvolvida tivesse um melhor desempenho para a questão de classificação e mapeamento de imagens, a camada de entrada apresenta sempre valores referentes ao padrão RGB de cor. Desta forma, estão presentes três neurônios na entrada da rede neural, onde cada um deles é responsável pelas tonalidades de vermelho, verde e azul.

Assim, as imagens submetidas para análise e processamento da rede neural são apresentadas pixel a pixel, viabilizando uma taxa de acerto de 98,73%, valor este que pode ser considerado alto perante outras ferramentas semelhantes. Já em [15] é descrita uma solução para classificação de sons, com foco na identificação de vozes humanas. O processo de reconhecimento é semelhante ao efetuado para reconhecimento de cores. Entretanto, para identificação do som digital são necessárias mais dimensões do que aquelas necessárias para as cores. Assim, a camada de entrada da rede neural é composta por nove neurônios, elevando consideravelmente a complexidade e o custo de processamento para a convergência dos resultados da rede neural. Apesar deste alto custo de processamento, a solução é capaz de associar os sons com um grau de acerto de 95,76%.

O controle dinâmico de recursos físicos do ambiente de processamento através de redes neurais artificiais apresenta crescimento significativo nos últimos anos, e se mostra presente em muitos estudos. Em [16] é descrita a utilização de uma rede neural do tipo (MLP – *Multi Layer Perceptron*) que tem por objetivo indicar a freqüência de CPU a serem configuradas, através do mecanismo DVFS, no ambiente para prover economia de energia e manutenção do nível de serviço. Foram mapeadas freqüências de CPU disponíveis para utilização no servidor *Web* e taxas de ocupação corrente da CPU para associação desses valores a um consumo de energia estimado. Esta estimativa se baseia na relação entre frequência de CPU e consumo de energia do *host*, a qual indica que quanto menor for a freqüência utilizada menor será o consumo de energia. A rede neural configurada para o trabalho é treinada para maximizar a taxa de ocupação da CPU e minimizar a freqüência de operação e potência consumida do *host*.

O estudo realizado em [17] utiliza uma rede neural com mapas auto-organizáveis para classificar estados de um servidor *Web* com base em uma série de medidas, como uso de CPU, número de pacotes recebidos na interface de rede, blocos lidos/escritos no disco rígido, etc. Ao total, foram utilizadas nove dimensões na entrada da rede neural, relacionadas às nove medidas coletadas do ambiente. Assim, a rede apresentou quatro estados principais de ocupação do servidor, que serviram de base construção de uma política de tolerância a falhas implementada pelo trabalho. Em [18] foi desenvolvida uma rede neural com uso de mapas auto-organizáveis para classificar estados referentes à utilização de memória em um servidor com ambiente Linux. A rede neural foi implementada utilizando como dados de entrada três indicadores relativos ao consumo de memória no ambiente. Após a fase de treinamento, a configuração dos mapas auto-organizáveis indicou seis estados possíveis de consumo de

memória. Com base na faixa de valores relacionada a cada estado foi implementado um módulo atuador para gerenciamento dos processos no ambiente Linux que substituiu a funcionalidade *OOM Killer*, que é nativa do *kernel* do Linux e responsável pala gerência de memória do ambiente de processamento.

1.3 Discussão das soluções apresentadas

A solução implementada por [7] trata de forma eficiente a manutenção da qualidade de serviço procedendo em média a redução de 18,6% do consumo de energia do ambiente. Conforme registros anteriores, o referido trabalho serviu de base para este estudo e é analisado com detalhes no Capítulo 3. Entretanto, há a possibilidade de incrementar a economia de energia alcançada pela solução mencionada efetuando melhorias no processo intervenção junto ao *cluster* de VMs, observando outros indicadores do ambiente como o próprio consumo de energia e a taxa de requisições submetidas aos servidores virtuais. Os trabalhos descritos em [9], [10] e [11] também propõem uma arquitetura de suporte a domínios de VMs e abordam o tema de gerência de recursos de um *cluster* de servidores virtuais utilizando como insumos para o processo de tomada de decisão métricas semelhantes como tempo de resposta da aplicação, carga de requisições submetidas, número de conexões abertas, etc. Entretanto, eles não se utilizam da informação de consumo de energia para compor as regras de intervenção no conjunto de servidores virtuais, e também não fazem uso de mecanismos inteligentes para definição de qual ação deve ser realizada e verificação de quando uma intervenção é necessária.

Os trabalhos apresentados em [16], [17] e [18] descrevem soluções desenvolvidas para controle e gerência de recursos computacionais através da utilização de redes neurais artificiais. Estas soluções se mostraram eficientes para a finalidade desejada, e as redes neurais utilizadas são capazes de modelar corretamente o cenário proposto em cada estudo. Porém, esses trabalhos não abordam indicadores relativos a aplicações suportadas pelo ambiente de processamento, efetuando uma classificação de estados apenas para recursos de *hardware* como blocos lidos do disco rígido, taxa de memória utilizada, etc. Uma justificativa para este tipo de tratamento é a dificuldade em efetuar o treinamento da rede neural perante métricas relacionadas às aplicações suportadas pela infra-estrutura computacional, pois se faz necessário ter um histórico de execuções das aplicações para observar seu comportamento e submeter os dados coletados para processamento da rede neural. Sem um ambiente de teste

real ou informações históricas do comportamento das aplicações torna-se inviável realizar um treinamento adequado para a rede neural, o que impede sua utilização para mapeamento e classificação dos processos desejados.

Como já existe um ambiente disponível para testes reais e uma arquitetura funcional desenvolvida em [7] disponíveis para utilização no presente trabalho, torna-se possível submeter para avaliação da rede neural indicadores relacionados ao funcionamento da aplicação, como tempo de resposta e carga de requisições submetidas, por exemplo. Desta forma, este estudo se utiliza desta abordagem para espelhar de forma mais completa o estado de operação do *cluster* de máquinas virtuais, provendo intervenções mais adequadas e eficientes no ambiente de processamento em busca de uma redução média do consumo de energia mais significativa do que aquela observada na solução original.

CAPÍTULO 2 VIRTUALIZAÇÃO E REDES NEURAIS

Neste capítulo são apresentados os conceitos básicos de virtualização de servidores e de redes neurais artificiais, ferramentas que foram utilizadas neste trabalho. Questões referentes ao funcionamento dos mecanismos implementados pela virtualização, tais como controle de acesso aos dispositivos físicos do servidor pelas máquinas virtuais, gerência das chamadas de sistema efetuadas por cada domínio virtual, economia de energia provida pelo uso de virtualização são abordadas. Também são apresentados os conceitos e funcionamento de uma rede neural artificial, com ênfase nos mapas auto-organizáveis, também chamados de redes de Kohonen, técnica esta escolhida para compor a solução desenvolvida neste trabalho.

2.1 Características e evolução da virtualização

Virtualização é um processo que provê de forma consistente o uso dos mesmos dispositivos físicos de um computador por várias entidades virtuais [19]. Esta consistência garante que as múltiplas instâncias virtuais hospedadas em um servidor físico estejam isoladas uma das outras, ou seja, cada máquina virtual (*Virtual Machine* – VM) opera de forma transparente e isolada perante as demais. A utilização da virtualização possibilita um ambiente mais flexível suportado pelo servidor físico, já que cada VM pode operar em um sistema operacional distinto e prover aplicações e serviços diferenciados em relação às outras. Assim, um único servidor físico pode se tornar um provedor heterogêneo de serviços, que podem ser disponibilizados por VM distintas, possibilitando a utilização de um quantitativo menor de máquinas e equipamentos em ambientes de processamento de dados.

As primeiras definições e implementações de máquinas virtuais surgiram em meados da década de 60, com o IBM S/370. Naquele projeto a IBM desenvolveu um modelo na qual cada máquina virtual era uma cópia fiel de uma máquina real, mas com uma capacidade de memória reduzida. Desta forma, o computador físico poderia ser dividido em vários computadores virtuais mais leves, possibilitando a segmentação de funcionalidades e serviços que estariam distribuídos pelas instâncias virtuais ao invés de permanecerem consolidados em um único ambiente [20]. Com o aprimoramento dos mecanismos de virtualização, as principais técnicas disponíveis nos dias atuais vão muito além da simples fragmentação de um ambiente único de processamento. A capacidade de suportar ambientes heterogêneos com configurações totalmente distintas entre si, e a gerência desses ambientes através de uma

entidade chamada de monitor de máquinas virtuais (*Virtual Machine Monitor* – VMM) estão dentre as principais funcionalidades utilizadas pelos usuários que optam pelo uso da virtualização.

Cabe ressaltar que foram executadas mudanças significativas nos processadores de forma a prover suporte para a virtualização. Dentre as principais mudanças, pode-se destacar a *Intel Virtualization Technology* (IVT) desenvolvida em 2005 pela Intel que trouxe adaptações do processador para as técnicas de virtualização [21], permitindo o funcionamento do processador como se existissem vários processadores paralelos, aumentando assim o desempenho do sistema operacional convidado. Já a fabricante de processadores AMD disponibilizou em 2006 a tecno*log*ia *AMD Virtualization* que visa diminuir a complexidade dos *software*s de virtualização transferindo diretamente para o *hardware* do processador algumas operações antes delegadas à camada de virtualização [22].

2.2 Técnicas de virtualização

As técnicas de virtualização apresentam algumas similaridades entre si, e diferem basicamente pela complexidade de implementação, suporte provido aos sistemas operacionais e controle de acesso aos dispositivos de *hardware* [23]. As três principais técnicas de virtualização consistem em: emulação, virtualização completa e paravirtualização, e são apresentadas de forma mais detalhada a seguir.

2.2.1 Emulação

Esta técnica implementa uma virtualização de *hardware*, inserindo através de *software* uma camada entre a aplicação e o sistema operacional. Conforme observado na Figura 3, o emulador funciona abstraindo das aplicações o *hardware* real da máquina, provendo uma camada virtual que simula o *hardware* necessário para as aplicações. Pode-se observar que tanto o emulador quanto o *hardware* virtual fazem parte da mesma camada, que opera sobre um sistema operacional e serve de base para as aplicações que se utilizam dos emuladores.

Figura 3 – Estrutura da camada de emulação

Através do *software* de emulação é realizada a transcrição das instruções de um processador real alvo para o processador virtual onde o emulador é executado. Desta forma, a aplicação que será suportada pela camada de emulação pode ser um sistema operacional distinto daquele que faz a interface com os dispositivos de *hardware* real da máquina, ou simplesmente uma aplicação que necessita de mecanismos de controle distintos daqueles providos pelo sistema operacional original do hospedeiro. Como exemplos desta técnica de emulação de *hardware* podem ser citados os consoles de jogos mais antigos, como Master System, Mega Drive, Atari, etc. que são executados em um computador, o Virtual PC da Microsoft que permite emular um sistema operacional MAC OS em uma janela do Windows XP conforme abordado em [24], e o VMWare em sua versão original que suporta uma variedade maior de sistemas operacionais para emulação como Linux, Windows, Solaris, dentre outros.

2.2.2 Virtualização completa

A virtualização completa é efetuada diretamente no sistema operacional hospedeiro. Assim, a camada de virtualização também funciona como uma abstração das aplicações e repassa as instruções ao sistema operacional hospedeiro, que por sua vez efetua o controle e a gerência dessas requisições junto ao *hardware*. Na Figura 4 pode ser verificado que acima da camada de virtualização completa podem ser executadas várias VMs, que podem possuir seus sistemas operacionais convidados independentes com suas respectivas aplicações e serviços. Entretanto, não é possível executar um sistema operacional convidado de *kernel* diferente do hospedeiro.

Figura 4 – Camada de Virtualização Completa

Nesta abordagem, cada entidade de máquina virtual simula o *hardware* completo para os sistemas operacionais instalados em cada VM. O OpenVZ [25] é um exemplo de *software* que implementa a técnica de virtualização completa.

2.2.3 Paravirtualização

Nesta técnica a camada de paravirtualização é o próprio sistema operacional modificado para este fim. Não há simulação explícita do *hardware*, e sim uma API específica disponibilizada através da modificação no *kernel* do sistema operacional para permitir chamadas ao *hardware*. Conforme indicado na Figura 5, a camada de paravirtualização faz a interface com o *hardware* da máquina e trabalha em conjunto com o monitor de máquinas virtuais, também chamado de *hypervisor*, que faz a gerência e o controle das chamadas efetuadas pelas máquinas virtuais.

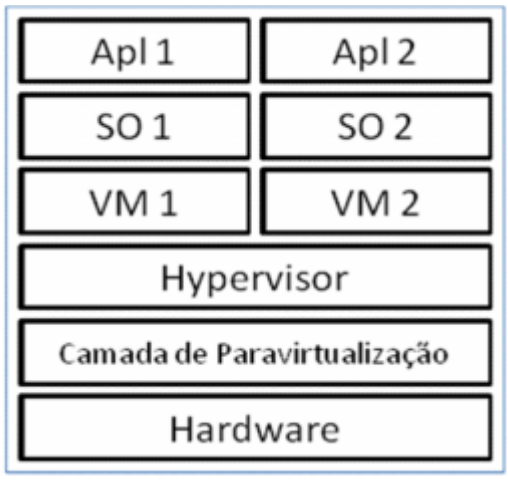


Figura 5 – Camada de paravirtualização

A diferença entre a técnica de virtualização completa e a paravirtualização consiste na forma como cada uma implementa a comunicação entre as VMs e o sistema operacional. Enquanto na virtualização completa há uma camada de interface entre o sistema operacional e as máquinas virtuais, na paravirtualização esta camada é próprio sistema operacional que é modificado para suportar o *hypervisor*, que é um VMM responsável pelo controle das requisições de cada VM. Dois exemplos desta técnica são o Xen [26], que possui uma versão construída em código aberto e consiste no principal *software* de virtualização utilizado pela comunidade acadêmica, e a recente versão do VMware construída para trabalhar com a paravirtualização. Em virtude da utilização do Xen na solução original que inspirou este estudo, serão apresentadas na seção a seguir as principais características da ferramenta.

2.3 A ferramenta de virtualização Xen

O Xen trabalha com a técnica de paravirtualização, onde o sistema operacional convidado executado em uma máquina virtual acredita que está em operação de forma isolada e interagindo de forma direta com o *hardware*, sem tomar conhecimento de que podem existir outras máquinas virtuais em operação no mesmo servidor hospedeiro. O Xen encarrega-se de organizar as requisições efetuadas pelas máquinas virtuais e repassá-las ao sistema operacional hospedeiro. Esta tarefa é efetuada por uma entidade chamada *hypervisor*. Como o Xen não faz a interpretação das requisições, apenas realizando o repasse das mesmas para o sistema operacional, ele consegue obter um bom desempenho para as aplicações que são executadas nas máquinas virtuais. Além de ser uma solução já consolidada no mercado, o Xen possui código fonte aberto permitindo que os pesquisadores possam analisá-lo e promover melhorias e contribuições.

A grande parte dos sistemas operacionais que trabalham com a arquitetura de processador X86 funciona em nível 0 (*ring level* 0), onde podem ser executadas qualquer instrução privilegiada e qualquer acesso de entrada/saída. A implementação do Xen altera o *kernel* do nível 0 para o nível 1, passando o próprio *hypervisor* a ser executado em nível 0. Essa operação é transparente para as aplicações, já que as mesmas são executadas em nível 3, que é o menos privilegiado de todos. Desta forma, o *hypervisor* é executado no nível 0, os sistemas operacionais convidados operam no nível 1 e as aplicações no nível 3. O nível 2 raramente é utilizado. No momento do boot de um computador que trabalha com esta solução, o *hypervisor* é carregado na memória do nível 0 e então inicia um *kernel* modificado no nível

1, que é chamado de dom0. Neste domínio dom0 é possível criar e destruir outros domínios, chamados de domU, além de efetuar migração dos mesmos entre diferente servidores hospedeiros físicos, dentre outras operações. Conforme já mencionado anteriormente, todas as chamadas dos sistemas operacionais convidados (domU) precisam obrigatoriamente passar pelo *hypervisor* (dom0).

A Figura 6 mostra o relacionamento entre os domínios virtuais criados, o *hypervisor* e o *hardware* físico. Os domínios virtuais (também chamados de máquinas virtuais ou domU) operam com seus respectivos sistemas operacionais convidados e cabe ao *hypervisor* controlar a alocação dos recursos reais do *hardware* através da simulação de um *hardware* virtual para os domU. Além de controlar as chamadas dos domínios virtuais para o *hardware* real, o *hypervisor* provê uma API para manipulação desses domínios conforme as necessidades do usuário. Para o estudo em [7] a Xen API foi modificada, com alterações e inclusões de novas funcionalidades relacionadas ao monitoramento do ambiente e principalmente à intervenção nos hospedeiros físicos e servidores virtuais. Os detalhes da nova API estão descritos no Capítulo 3, onde são apresentados o sistema proposto e a nova política de reconfiguração.

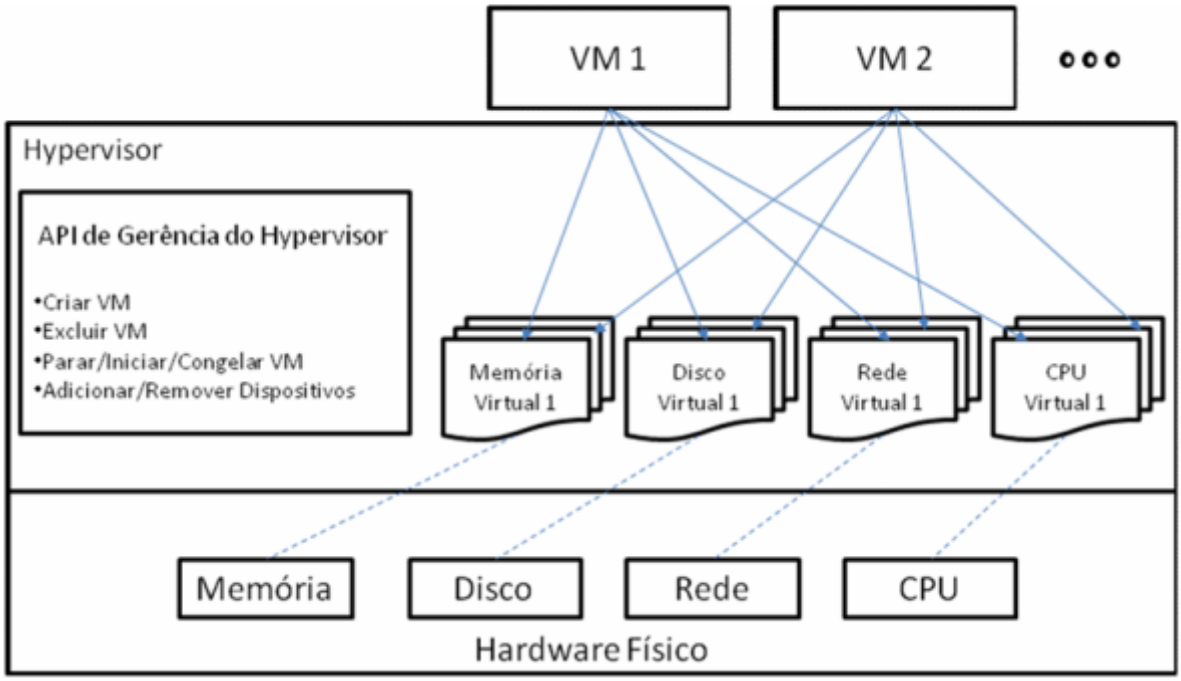


Figura 6 – Arquitetura implementada pelo Xen

Uma tarefa importante realizada pelo *hypervisor* é a simulação do *hardware* real de uma forma simplificada para os domínios virtuais. Assim, o mapeamento do *hardware* físico para o *hardware* virtual é efetuado pelo *hypervisor* de forma a abstrair para as máquinas

virtuais eventuais singularidades dos dispositivos físicos. Por exemplo, não importa qual seja a placa de rede instalada fisicamente no servidor, pois o *hypervisor* expõe para o domínio virtual apenas uma interface de rede genérica por onde o mesmo irá realizar sua comunicação externa. Assim, por causa desta característica de implementação, os *drivers* dos dispositivos físicos necessitam estar instalados e configurados apenas do dom0.

O compartilhamento dos recursos de *hardware* físico é realizado pelo *hypervisor* através do fornecimento de uma porção desses recursos para os domínios virtuais. De uma forma geral, o *hypervisor* não dispõe toda a capacidade de recursos para um determinado domínio, e sim faz uma divisão dos recursos disponíveis para as máquinas virtuais criadas. É possível até determinar ao *hypervisor* que algumas VMs sejam privilegiadas em relação às demais nas regras de acesso ao *hardware*, o que pode se mostrar uma alternativa interessante perante as características das aplicações que são executadas em cada máquina virtual.

2.4 Redes neurais artificiais

As redes neurais artificiais consistem em uma técnica utilizada para solucionar problemas de inteligência artificial. Esta abordagem se caracteriza pelo uso de um grupo de neurônios artificiais interconectados, e se utiliza de um modelo matemático baseado no comportamento do cérebro humano para processamento de informações [27]. A premissa básica é realizar o processamento de informações conforme o modelo de organização de neurônios do cérebro humano. Como o cérebro é capaz de aprender e tomar decisões baseadas em aprendizagem, as redes neurais artificiais possuem esta mesma característica. Assim, uma rede neural pode ser interpretada como um esquema de processamento capaz de armazenar conhecimento baseado em aprendizagem (treinamento) e disponibilizar este conhecimento para uma entidade externa, que neste estudo se materializa no sistema proposto para gerenciamento do ambiente computacional.

Para que uma rede neural possa responder corretamente às informações submetidas faz-se necessário a realização de uma etapa chamada de treinamento, que tem por objetivo definir uma sistemática para processar adequadamente o conjunto de dados fornecidos. Desta formas, a rede neural é capaz de extrair dinamicamente regras e características básicas a partir de dados reais, diferindo dos sistemas especialistas onde é necessária implementação de um conjunto de regras rígidas pré-estabelecidas. De uma forma geral, os tipos de rede neural diferem fundamentalmente em relação à metodologia de treinamento, o supervisionado e o

não supervisionado. Os detalhes e o funcionamento das duas metodologias são apresentados na seção seguir.

2.4.1 Treinamento de uma rede neural

Apesar de diferirem por suas formas de treinamento, as redes neurais artificiais possuem uma arquitetura comum, onde são verificadas até três camadas de neurônios: (i) camada de entrada, (ii) camada escondida ou intermediária e (iii) camada de saída. Em redes neurais com treinamento supervisionado as três camadas estão presentes, já naquelas com treinamento não supervisionado são verificadas apenas as camadas de entrada e escondida, estando ausente a camada de saída. Assim como no cérebro humano, em uma rede neural artificial a conexão entre os neurônios é chamada de sinapse. Para cada conexão são atribuídos valores que são denominados de pesos sinápticos, que serão os responsáveis pela armazenagem do conhecimento obtido pela rede após sua etapa de treinamento.

A Figura 7 apresenta uma rede neural artificial formada por uma camada de entrada com três neurônios, uma camada escondida com cinco neurônios e uma camada de saída com um único neurônio. As arestas representam as conexões entre os neurônios, onde cada uma possui um valor determinado, o chamado peso sináptico. Os valores presentes nos pesos sinápticos indicarão a intensidade de conexão entre dois neurônios específicos. Pode-se observar que a camada de entrada é responsável por receber as informações submetidas à rede, e a camada de saída é encarregada de retornar ao ambiente externo o resultado do processamento de cada série de valores de entrada, onde no exemplo ilustrado cada série é constituída de três valores distintos (um valor específico para cada neurônio presente na camada de entrada).

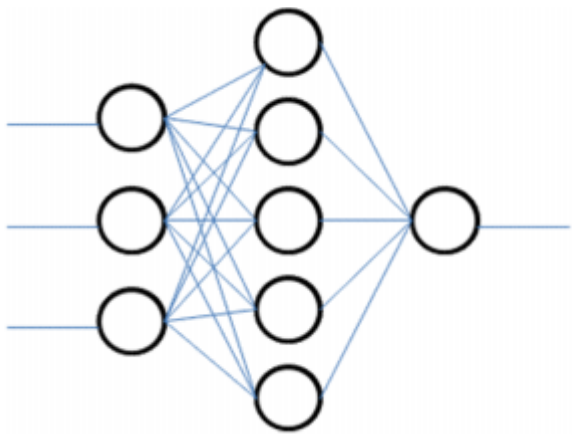


Figura 7 – Exemplo de uma rede neural artificial

No treinamento supervisionado, os valores de entrada e de saída desejada são fornecidos para a rede, ou seja, os valores de saída já são conhecidos e são considerados os valores alvo da rede. Desta forma, a tarefa da rede neural será apresentar em sua camada de saída os valores mais próximos possíveis daqueles desejados. Um agente externo então disponibiliza os valores fornecidos à camada de entrada da rede neural, que por sua vez encaminha esses valores para a camada escondida. Os neurônios da camada escondida efetuam o processamento das informações de acordo com as regras e especificações atribuídas à rede e produzem os valores a serem indicados na camada de saída. Assim, caberá ao agente externo observar se os valores indicados na camada de saída da rede estão condizentes com o resultado esperado, decretando assim se o treinamento realizado obteve sucesso ou não.

Em geral este tipo de treinamento utiliza o algoritmo de retropropagação do erro verificado na camada de saída. Neste algoritmo um valor de erro é calculado com base na comparação entre a resposta desejada pelo agente externo que é indicada para a rede e a resposta real produzida pela rede. O valor de erro é propagado de volta para a estrutura neural, ajustando os pesos sinápticos dos neurônios. Este procedimento é realizado inúmeras vezes até que os pesos ajustados da rede consigam prover uma resposta real satisfatória, ou seja, com uma taxa de erro praticamente desprezível. A rede neural do tipo Perceptron Multicamadas (*Multi-Layer Perceptron* – MLP) é exemplo mais tradicional que se utiliza do treinamento supervisionado com retropropagação do erro.

No treinamento não supervisionado, também chamado de adaptativo, somente os valores de entrada são fornecidos para a rede. Desta forma, não há valores alvo de saída indicados previamente para a estrutura. A própria rede neural define os critérios utilizados para agrupar e organizar os valores de entrada que serão processados em sua camada escondida com base na função de atualização dos pesos sinápticos definida pelo usuário. Este procedimento geralmente é chamado de adaptação ou auto-organização. A rede possui habilidade de formar representações internas para codificar as características da entrada, criando automaticamente novas classes conforme elas forem surgindo no decorrer do treinamento. Por fim, após o treinamento a rede neural terá efetuado uma classificação de classes (processo também denominado de *clustering*) de acordo com os padrões observados nos valores de entrada. Os mapas auto-organizáveis, utilizados na solução presente neste trabalho, se utilizam deste tipo de treinamento.

Independente da metodologia utilizada a fase de treinamento de uma rede neural pode ser expressa de forma simplificada através do algoritmo descrito a seguir:

1. Uma série de valores de entrada (X_1, X_2, X_3,...,Xn) é apresentada a camada de entrada da rede neural;
2. Os pesos sinápticos são ajustados para capacitar a rede a identificar o padrão fornecido pela série de valores;
3. As etapas 1 e 2 são repetidas até que todas as séries de valores de entrada sejam fornecidas para a rede;
4. Os procedimentos de 1 a 3 são repetidos exaustivamente até que seja encontrada uma configuração de pesos sinápticos capaz de identificar dentro da margem de erro estabelecida todos os padrões fornecidos.

Cabe ressaltar que a eficácia de uma rede neural consiste na aptidão da mesma em reconhecer padrões de comportamento que não foram apresentados em sua etapa de treinamento. Esta característica é denominada de capacidade de generalização da rede neural. De uma forma geral, esta capacidade só é obtida através de um treinamento intenso com alto custo computacional de processamento. É comum que cada iteração de treinamento, representado pelas etapas 1, 2 e 3 do algoritmo acima mencionado seja repetida milhares ou até milhões de vezes para que a rede assuma uma capacidade de generalização satisfatória.

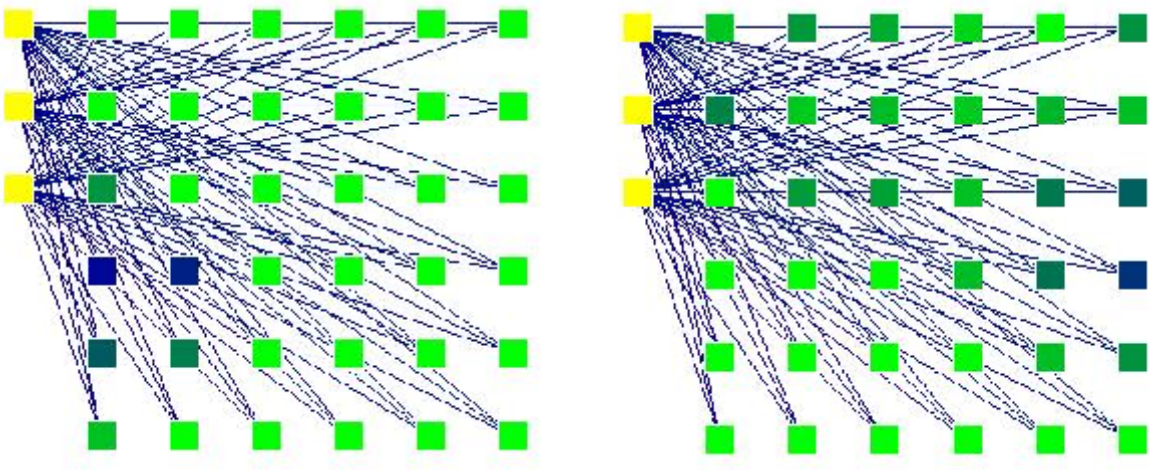
Na seção a seguir serão descritos os detalhes de funcionamento de uma rede neural com mapas auto-organizáveis, indicando os algoritmos e modelos matemáticos utilizados pela rede e suas principais configurações possibilitadas através da variação de parâmetros internos do seu funcionamento.

2.4.2 Mapas auto-organizáveis

As redes neurais artificiais que utilizam mapas auto-organizáveis, ou simplesmente SOM (*Self Organizing Maps*), se utilizam de treinamento não supervisionado baseadas em grades de neurônios artificiais geralmente formadas por duas dimensões. Esta técnica foi desenvolvida por Teuvo Kohonen e muitas vezes é chamada de mapas de Kohonen [28]. Embora de uma forma geral nas redes SOM os neurônios presentes na camada escondida estejam dispostos em uma grade unidimensional ou bidimensional, existem estudos que implementam grades com um maior número de dimensões conforme pode ser observado em [29] e [30]. Entretanto este tipo de abordagem não é muito utilizado, pois uma das principais funcionalidades da rede SOM é representar as múltiplas informações de entrada em um ambiente de dimensões menores.

Um mapa auto-organizável é caracterizado pela formação de um mapa topográfico que representa o estímulo dos neurônios a uma série de padrões de entrada, onde as localizações espaciais dos neurônios da camada escondida e suas respostas aos padrões de entrada apresentados à rede indicam os atributos inerentes aos dados apresentados à rede. Esta característica permite que diferentes padrões de entrada estimulem regiões distintas do mapa, possibilitando assim observar padrões de entrada que possuam atributos comuns, pois esses padrões irão estimular regiões próximas no mapa. Desta forma, a auto-organização dos neurônios na camada escondida perante os valores apresentados na camada de entrada originou o nome de mapas auto-organizáveis para as redes neurais que se utilizam desta abordagem.

A Figura 8 descreve o exemplo de uma rede neural com mapa auto-organizável com a presença de três neurônios na camada de entrada (grifados na cor amarela) e uma grade bidimensional 6x6 de neurônios na camada escondida. Conforme já mencionado anteriormente, é possível observar que cada neurônio da camada de entrada está conectado a todos os neurônios da camada escondida. Cabe ressaltar que os valores de entrada da rede (a) são diferentes dos valores de entrada da rede (b). Assim, é possível constatar que cada série de valores de entrada distinta estimula regiões topologicamente diferentes no mapa, o que pode ser constatado pela coloração das regiões da grade de neurônios, onde a cor azul representa os neurônios com um nível elevado de excitação e a cor verde um nível pequeno de excitação.



(a) **(b)**

Figura 8 – (a) Ativação dos neurônios para o padrão de entrada 1

Figura 8 – (b) Ativação dos neurônios para o padrão de entrada 2

Cada nó da grade possui uma posição topológica específica representada por uma coordenada (x, y), caso a grade de neurônios seja bidimensional. Além da coordenada cartesiana, cada neurônio da grade contém um vetor de pesos sinápticos de dimensão igual ao vetor formado pela série de valores de entrada. No caso do exemplo indicado na Figura 8, cada série de valores é formada por um vetor de três dimensões $x = (x_1,x_2,x_3)$. Desta forma, cada neurônio da grade contém um vetor de pesos sinápticos de três dimensões $w = (w_1,w_2,w_3)$.

A formação do mapa está relacionada aos vetores de pesos sinápticos associados a cada neurônio da camada. Assim, antes da execução do algoritmo de treinamento da rede que irá gerar como produto final um mapa auto-organizável associado a todos os valores de entrada apresentados, os valores vetores de pesos sinápticos devem ser iniciados. A forma mais comum de se fazer esta iniciação é atribuir aleatoriamente valores dentro da faixa de [-1,1] para cada neurônio da grade. Este procedimento é realizado para evitar que a rede seja induzida a alguma organização prévia antes da etapa de treinamento, fato este que poderia impedir a correta formação do mapa topográfico final após a etapa de treinamento.

Após a inicialização dos pesos sinápticos indicada acima a fase de treinamento da rede é iniciada e um vetor de entrada é escolhido do conjunto de dados de treinamento e apresentado para a grade de neurônios na camada escondida. Todos os pesos sinápticos dos neurônios da camada escondida são calculados para determinar o neurônio que possui o vetor mais similar ao vetor dos valores de entrada. Esta seleção é feita através do cálculo da distancia euclidiana entre o vetor de pesos sinápticos de cada neurônio e o vetor de entrada, conforme Equação 2. Aquele neurônio que tiver a menor distância euclidiana será denominado de BMU (*Best Matching Unit*);

$$dist_j = \sqrt{\sum_{i=1}^{n} (x_i - w_i)^2} \qquad (2)$$

Com a escolha do neurônio vencedor na iteração, é calculado o raio de vizinha topológica do BMU. Os neurônios localizados dentro deste raio são considerados vizinhos do BMU e terão os seus pesos sinápticos ajustados na etapa posterior denominada de adaptação sináptica. Geralmente o valor inicial do raio de vizinhança é definido como igual ao tamanho da grade, e torna-se decrescente no decorrer do treinamento. Este procedimento tem por objetivo que ao final do treinamento as regiões classificadas indicadas no mapa sejam consistentes, exibindo um distanciamento satisfatório de outras regiões que representam

classificações distintas. Esta redução gradual o raio de vizinhança do BMU é regida pela Equação 3, onde $\sigma 0$ representa o raio σ no tempo t_0 e λ é uma constante de tempo. A variável t representa um determinado instante durante a fase de treinamento.

$$\sigma(t) = \sigma 0 * e^{(-\frac{t}{\lambda})} \tag{3}$$

Definido o raio de vizinhança do BMU, todos os neurônios que estão inseridos na referida área, inclusive o próprio neurônio vencedor, terão seus pesos sinápticos ajustados. A Equação 4 descreve como a atualização do peso sináptico é realizada, onde t representa o instante atual, $\eta(t)$ é o parâmetro de taxa de aprendizagem variável no tempo que decresce gradualmente com o aumento do tempo t, e $\Theta j(t)$ é uma função variável no tempo que determina o grau de aprendizagem do neurônio j baseada na distância relativa com o BMU.

$$W_j(t+1) = W_j(t) + \eta(t) * \Theta_j(t) * (V(t) - W_j(t)) \tag{4}$$

A função que define a taxa de aprendizagem é apresentada na Equação 5 e apresenta um comportamento de decaimento exponencial idêntico a função descrita na Equação 3. O objetivo desta abordagem é que a taxa de aprendizado inicie com um valor padrão estipulado durante a configuração inicial da rede e apresente valores próximos a zero na fase final de treinamento. Além da influência da taxa de aprendizagem na atualização dos pesos sinápticos, a distância entre o neurônio j e o BMU também é considerada. Esta relação é especificada na Equação 6, onde $dist_j$ corresponde à distância euclidiana quadrática entre o neurônio j e o BMU, e $\Theta j(t)$ é o raio obtido pelo cálculo da vizinhança topológica conforme descrito na Equação 3.

$$\eta(t) = \eta 0 * e^{(-\frac{t}{\lambda})} \tag{5}$$

$$\Theta_j(t) = e^{(-\frac{distj^2}{2\sigma^2(t)})} \tag{6}$$

Na seção a seguir é apresentada a ferramenta de simulação de rede neural utilizada neste trabalho, assim como os valores utilizados para construção exemplo da rede neural artificial utilizada como exemplo na Figura 8 da presente seção.

2.4.3 A ferramenta SNNS (*Stuttgart Neural Network Simulator)*

A ferramenta SNNS (*Stuttgart Neural Network Simulator)* [31] é um simulador de rede neural artificial desenvolvido em 1989 pela Universidade de Stuttgart. O projeto de implementação do simulador buscou criar um ambiente flexível de plataformas e eficiente para pesquisas relacionadas às redes neurais artificiais. Em virtude de sua configuração simplificada e capacidade elevada de processamento, esta ferramenta está presente em um grande número de trabalhos acadêmicos que realizam estudos sobre redes neurais artificiais.

O simulador SNNS é formado por quatro módulos distintos, com destaque para o *kernel*, que é responsável pelas operações das redes neurais construídas através da utilização da ferramenta e uma GUI (*Graphical User Interface*), que foi construída sobre as especificações do *kernel* e disponibiliza uma plataforma gráfica para criação, manipulação e configuração das redes neurais desejadas. Cabe ressaltar que todas as funcionalidades disponíveis na interface gráfica também podem ser utilizadas através de uma entidade de processamento *batch* denominada de *batchman*, onde as manipulações das redes neurais são realizadas através de linhas de comando.

Os mapas auto-organizáveis estão presentes na ferramenta SNNS. Conforme já abordado na seção 2.4.2, o simulador SNNS implementa duas camadas de neurônios interligadas entre si, a camada de entrada onde serão propagadas as informações inseridas na grade de neurônios e a camada escondida responsável pelo processamento das regras especificadas para a rede neural. Para configuração do treinamento de uma rede neural com mapas auto-organizáveis a ferramenta SNNS possibilita a indicação cinco parâmetros para a função de aprendizagem (treinamento), denominada de “Kohonen”:

- Peso de adaptação: o peso inicial de adaptação *h(0)* da rede pode variar entre valores de 0 a 1. Esta taxa determina a força de adaptação dos pesos sinápticos em cada iteração da fase de treinamento, quanto maior for a taxa de aprendizagem

menos iterações serão necessárias para atingir a convergência para o comportamento desejado da rede neural. De uma forma geral, utiliza-se uma taxa alta de aprendizagem para experimentos que possuam poucos dados disponíveis para serem fornecidos a rede neural na fase de treinamento. Entretanto, uma taxa elevada de aprendizagem pode "viciar" a rede construída, inibindo a sua capacidade de generalização;

- Raio de adaptação: O raio inicial de adaptação *r(0)* corresponde ao alcance da adaptação em relação à vizinhança do neurônio vencedor da iteração. Assim, todos os neurônios da grade que estão dentro do raio de vizinhança corrente terão seus pesos sinápticos adaptados pela função de aprendizagem. Os valores para definição do raio de vizinhança inicial podem variar entre 1 e o tamanho total do mapa;
- Taxa de decaimento do peso de adaptação: Para cada iteração da fase de treinamento o peso de adaptação será decrescido de um valor definido através da Equação 4 já apresentada na seção 2.4.2. Este decaimento é controlado através do fator de decaimento *mult_H* definido por: $h(t+1) = h(t) * mult\_H$;
- Taxa de decaimento do raio de adaptação: A cada apresentação de uma série de valores de entrada durante a fase de treinamento o raio de adaptação será diminuído de um valor definido através da Equação 3 já apresentada na seção 2.4.2. Este decaimento é realizado através do fator de decaimento *mult_R* definido por: $r(t+1) = r(t) * mult\_R$;
- Tamanho horizontal da grade de neurônios: É a quantidade de neurônios presente na coordenada horizontal *x* da camada escondida.

Além da configuração de parâmetros acima da função de aprendizagem "Kohonen", o simulador SNNS disponibiliza a função "Kohonen_Weights" para inicialização aleatória dos valores dos pesos sinápticos, e as funções "Act_Euclid" e "Kohonen_Order" que implementam as Equações 2 e 6, respectivamente, da seção anterior.

A Figura 9 descreve um *script* para de treinamento de uma rede neural SOM na ferramenta SNNS. Os comandos descritos no código apresentado são interpretados pelo *batchman* e processados pelo *kernel* da ferramenta. A rede neural composta por 3 neurônios na camada de entrada e por uma grade 6x6 de neurônios na camada escondida, que é apresentada na Figura 8 da seção 2.4.2, está especificada no arquivo "exemplo.net". As séries

de valores de entrada utilizada na etapa de treinamento estão descritas no arquivo "exemplo.pat". Ambos os arquivos são carregados pela ferramenta nas primeiras linhas do *script,* linha 1 para o arquivo da rede neural pré-construída e linha 2 para os valores de treinamento, de forma que os parâmetros de treinamento definidos na seqüência possam ser aplicados na rede neural.

```
1     loadNet("exemplo.net")
2     loadPattern("exemplo.pat")
# Define a função para iniciar aleatoriamente os valores dos pesos sinápticos
4     setInitFunc("Kohonen_Weights", 1.0, -1.0)
# Define a função de aprendizagem, como os parâmetros h(0)=0.9, r(0)=3.0,
# mult_H=0.98, mult_R=0.98 e x=6.0
5     setLearnFunc("Kohonen",0.9, 3.0, 0.98, 0.98, 6.0")
# define a função de ativação de vizinhança
6     setActivFunc ("Act_Euclid")
# define a função de atualização dos pesos sinápticos
7     setUpdateFunc("Kohonen_Order")
# inicia os pesos sinápticos conforme função definida na linha 4
8     initNet()
9     setPattern("xor.pat")
10    for i := 1 to 40000 do
11     trainNet()
12     print(MSE)
13    endfor # São definidas 40000 iterações de treinamento
14    setPattern("xor.pat")
15    saveResult ("xor-saida.txt")
16    saveNet("xor-trained.net")
```

Figura 9 - Script para treinamento de uma rede neural na ferramenta SNNS

CAPÍTULO 3 SISTEMA PROPOSTO

No presente capítulo é descrita a política de reconfiguração original, que serviu de base para este trabalho. Também é discutido o processo de validação da rede neural de mapas auto-organizáveis inserida na nova arquitetura, além da nova política de reconfiguração que sobrepõe a política original para gerência de recursos no *cluster* de servidores virtuais. A definição da nova política é baseada em testes de processamento com uma carga real, possibilitando a observação dos padrões de comportamento do *cluster* perante uma situação real de submissão de requisições. Para um melhor entendimento das funcionalidades desenvolvidas neste trabalho, apresentamos a seguir as premissas que nortearam o desenvolvimento deste trabalho.

3.1 Premissas e diretrizes da proposta

Conforme mencionado no Capítulo 2, a presente proposta tem por base a arquitetura e a API desenvolvidas por [7] onde foi obtida uma redução significativa do consumo de energia do *cluster* com a manutenção da qualidade de serviço. Entretanto, a seqüência das possíveis intervenções realizadas pelo gerenciador é rígida, restringindo as possibilidades de controle, levando, em alguns casos, a perdas de qualidade de serviço.

Avaliando a arquitetura original mais profundamente, verificamos alguns pontos que poderiam ser aprimorados, por exemplo: a rigidez nas possíveis intervenções implica em ações pouco eficientes em determinados cenários. Esta situação pode ser ilustrada quando o tempo de resposta medido viola as restrições de operação. Por definição, o sistema original seguirá o fluxo de intervenções pré-definido, escalonando as ações de entrega de recursos ao ambiente até que o tempo de resposta volte a apresentar valores abaixo do limite estipulado. Neste caso, seria mais interessante uma reconfiguração de ambiente mais arrojada, disponibilizando a quantidade necessária de recursos de uma única vez, já que o estado corrente de funcionamento do *cluster* é crítico, pois apresenta violação das restrições de operação.

A diretriz da presente proposta consiste em diferenciar as intervenções junto ao *cluster* de servidores virtuais de acordo com estados de operação classificados pela rede neural. A partir desta classificação, o processo de reconfiguração torna-se dependente apenas da situação corrente de funcionamento do sistema, pois cada estado classificado possui um uma

estratégia específica de intervenção definida. Assim, as ações de adaptação do sistema junto às VMs tornam-se mais flexível e seus resultados mais eficientes perante os seguintes aspectos: (i) economia de energia do conjunto de servidores físicos e (ii) tempo de resposta das requisições submetidas ao *cluster* de servidores virtuais.

Para construção das funcionalidades propostas neste trabalho, a arquitetura original já implementada no trabalho de Marcus Vinícius Azevedo é utilizada como base. Dentre as principais atividades realizadas no presente estudo e adicionadas ao sistema original para implementação da nova política de reconfiguração do *cluster* de servidores virtuais *web*, podemos destacar:

- Definição de estados de operação observados no *cluster*, conforme os valores medidos de (i) taxa de requisições submetidas aos servidores *Web* hospedados nas VMs, (ii) tempo de resposta do servidor *frontend* e (iii) potência consumida dos servidores hospedeiros;
- Adequação das funções desenvolvidas na API original para que as mesmas possam ser utilizadas de uma maneira mais eficiente perante os diversos estados do *cluster*, aumentando assim o dinamismo da solução, pois as intervenções não seguirão um roteiro rígido de operações.

A classificação dos estados de operação do *cluster* segundo as três medidas mencionadas acima é obtida através da utilização de uma rede neural artificial do tipo mapas auto-organizáveis, técnica esta que foi abordada em detalhes no Capítulo 2. Os parâmetros utilizados nesta classificação e os detalhes dos procedimentos realizados estão descritos mais adiante na Seção 3.3. Com base na classificação fornecida pela rede neural, cada estado de operação é associado a funções da API que realizam intervenções no *cluster*. Esta associação é realizada através de testes unitários para cada estado de operação classificado, possibilitando assim indicar quais ações são mais apropriadas para cada cenário, ou seja, quais delas fornecem resultados mais satisfatórios para as métricas de qualidade de serviço e consumo de energia do *cluster*. O mapeamento das funções da API perante as classificações obtidas do *cluster* de VMs também estão indicados na Seção 3.3.

3.2 Arquitetura e API originais

A arquitetura e a API originalmente desenvolvidas se utilizam de módulos independentes que fazem monitoração e intervenção no *cluster* de VMs que suportam os servidores virtuais com base nas regras definidas previamente nas políticas de reconfiguração. Para monitoração do ambiente e execução dos mecanismos de reconfiguração há os elementos Coletor e Atuador, respectivamente. Esses elementos executam seus procedimentos através da API, e estão subordinados ao elemento central da arquitetura denominado de Controlador, que é responsável por armazenar as configurações do *cluster* garantir a qualidade de serviço da aplicação suportada pelo *cluster* de VMs. A implementação da arquitetura é apresentada no diagrama da Figura 10, seguida de uma descrição dos componentes.

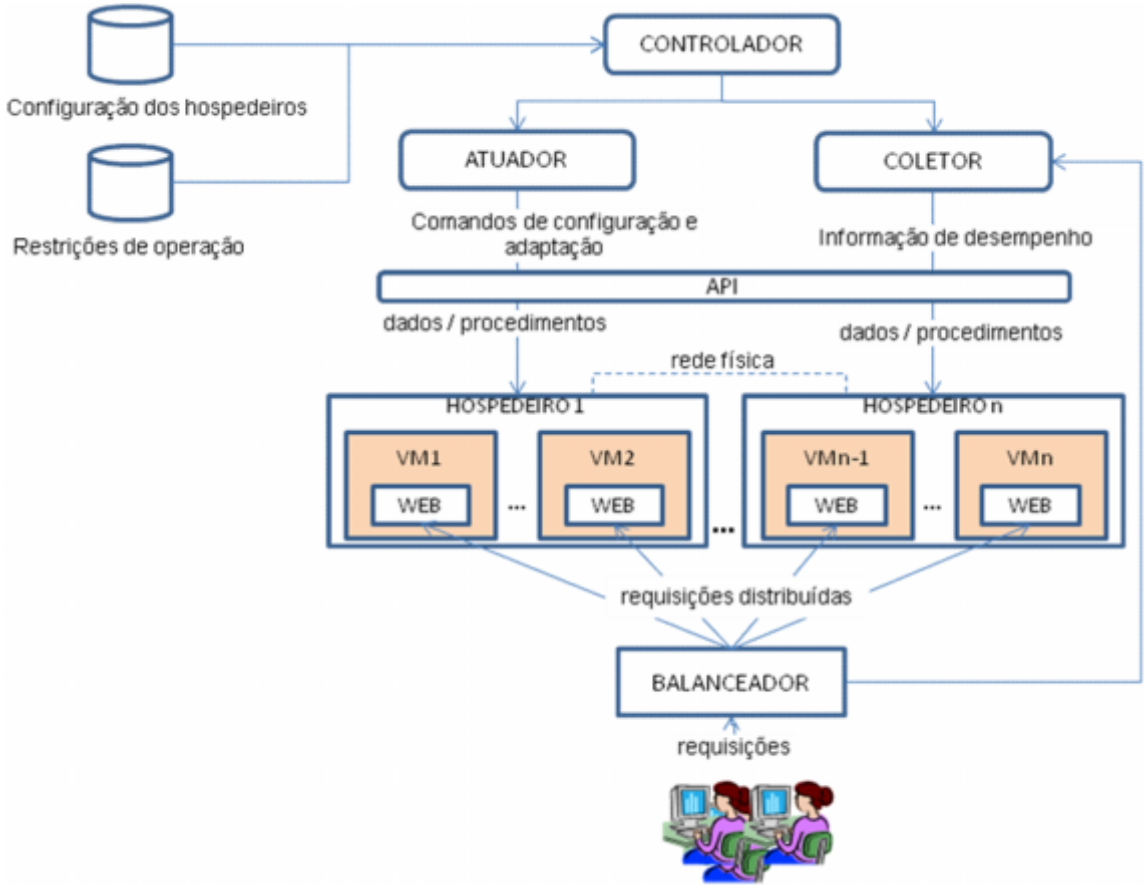


Figura 10 – Arquitetura implementada

Controlador: Elemento central da arquitetura. Inicialmente ele é carregado com as informações específicas dos *host*s (nome, endereço MAC, quantidade de processadores, freqüência, etc.) e do Balanceador (nome, URL, etc.), além das restrições de operação da aplicação *Web* (tempo de resposta máximo admitido). Durante sua execução verifica periodicamente o cumprimento das restrições de operação, com base nas informações

obtidas do Coletor, e aciona o Atuador para reconfigurar o ambiente adequadamente se necessário. O Controlador é o responsável por estabelecer a conexão inicial com cada *host* via API e mantém estas conexões para serem utilizadas pelo Atuador e Coletor posteriormente.

Atuador: Elemento responsável por efetivamente atuar nos *host*s. Utiliza a API para realizar operações para ativar ou desativar *host*s, aumentar ou diminuir a freqüência da CPU e ligar ou desligar processadores (ou núcleos). O Atuador também é responsável por migrar VMs quando solicitado.

Coletor: Este elemento, que também utiliza a API, obtém as informações dinâmicas dos *host*s (utilização de CPU, frequência atual, lista e quantidade de VMs hospedadas, etc.) e do Balanceador (atual tempo de resposta da aplicação *Web*). O Controlador aciona o Coletor periodicamente, em ciclos de medição.

API: Fornece uma camada de alto nível para que os elementos Controlador, Atuador e Coletor possam atuar sobre os *host*s e VMs tanto para coleta de informações quanto para reconfigurações.

Clientes: Representa o grupo de usuários finais do serviço provido, no caso, uma aplicação hospedada em servidores *Web*.

Balanceador: Este elemento é o responsável por distribuir as requisições dos Clientes para os servidores virtuais *Web*, provendo desempenho e escalabilidade.

Máquinas Virtuais: Elementos que residem nos *host*s. Cada VM suporta um servidor *Web* que recebe requisições distribuídas pelo Balanceador.

Servidores Físicos: Elementos responsáveis por hospedar as VMs, fornecendo de maneira compartilhada acesso aos recursos físicos (CPU, memória, disco e rede). Um desses elementos no conjunto será chamado de *host* principal, no qual ficarão armazenadas as imagens de disco das VMs compartilhadas via NFS. O hospedeiro principal será o único *host* que nunca será desligado para poder suportar todas as VMs do ambiente em situações de baixa utilização de recursos.

O funcionamento da arquitetura descrita acima é iniciado através da interpretação de dois arquivos XML, um contendo informações da configuração inicial dos *host*s e do Balanceador, e outro contendo as restrições de operação, entre elas o tempo de resposta máximo admitido para a aplicação instalada nos servidores virtuais *Web*. Em seguida, usando

a API, o Controlador abre as conexões com os *host*s gerenciados e ativa o Coletor, que inicia um ciclo de medição. Após o término do ciclo de medição, o Controlador obtém informações atualizadas sobre os *host*s e sobre a aplicação *Web*, e avalia se o tempo de resposta está acima ou abaixo do máximo admitido. Dependendo da avaliação uma rotina de reconfiguração é iniciada, e ao término desta um novo ciclo de medições começa.

Para o presente trabalho os módulos Controlador e Atuador são alterados. É inserido no Controlador toda a lógica e o processamento da rede neural, sobrepondo assim a política de reconfiguração original, cabendo agora a rede neural fornecer em sua interface de saída a classificação do estado corrente de operação do *cluster* conforme as medições coletadas do ambiente. É esta classificação de estado que indica a intervenção no *cluster* mais adequada em cada ocasião. Já o Atuador continua a ter os tipos de intervenções ditados pelo Controlador, mas tem suas ações ajustadas em relação à versão original, pois foi observado que determinadas intervenções são inócuas junto ao *cluster* em certas situações.

Na política de reconfiguração original realizada pelo Controlador a indicação de que o tempo de resposta da aplicação está acima do limite indicado nas restrições de operação, aciona o Atuador para que este reconfigure servidores hospedeiros, adicionando recursos aos mesmos ou distribuindo VMs entre eles. Caso o tempo de resposta esteja dentro ou abaixo das restrições de operação, o Atuador também é acionado para reconfigurar o ambiente só que, desta vez, com a finalidade de economizar energia, retirando recursos ou consolidando VMs, obedecendo as restrições de manutenção da qualidade de serviço. Os tipos de intervenção implementados na API junto ao *cluster* de máquinas virtuais estão descritas a seguir:

- Alteração da freqüência da CPU do servidor hospedeiro: Esta é a primeira intervenção a ser realizada para reconfiguração do *cluster*, produzindo o incremento ou retirada de recursos do ambiente dependendo do cenário observado. Caso a utilização de CPU do hospedeiro esteja acima do valor limite pré-definido, o Controlador aciona o Atuador para aumentar a freqüência de operação do processador. Por outro lado, se a utilização de CPU estiver abaixo do valor mínimo estipulado, o Atuador será acionado pelo Controlador para decrementar a freqüência de operação da CPU.
- Ativação/desativação de núcleos do processador: Este procedimento é realizado quando a freqüência de operação da CPU dos servidores físicos já se encontra em seu valor máximo ou mínimo permitido. Assim, para adição de recursos no ambiente, o Atuador irá intervir ligando mais núcleos nos servidores hospedeiros que estiverem

com freqüência de CPU já no valor limite, caso ainda existam núcleos que possam ser ativados. Caso o objetivo seja a retirada de recursos do ambiente, o Atuador tenta então desligar núcleos nos servidores físicos que estiverem com o valor mínimo de freqüência de CPU.

- Migração de máquinas virtuais entre os servidores hospedeiros: Caso não seja possível realizar os dois procedimentos anteriores, o Atuador busca um servidor físico com recursos disponíveis para fazer a migração de uma VM e efetua a migração, se possível. Para o cenário de retirada de recursos do ambiente o Atuador tenta consolidar todas as VM em um único hospedeiro. Se a intervenção tiver como objetivo a adição de recursos, o Atuador distribui as VM entre os hospedeiros visando uma menor competição por recursos de *hardware* entre as VM.

- Ativação/desativação de servidores físicos hospedeiros: Se não existirem *host*s com recursos disponíveis para receber a VM a ser migrada no caso anterior, o Atuador, ainda, ativa *host*s que estiverem em estado de espera para permitir a migração. Para o caso de economia de recursos, se um servidor físico hospedeiro tiver sua última VM migrada para outro servidor, o mesmo é desativado (inserido no modo de espera).

Cabe ressaltar que além das quatro funcionalidades principais descritos acima, a API também dá suporte às funções de monitoramento do *cluster*, que são geridas pelo módulo Coletor. Este módulo é responsável por executar as medições periódicas do *cluster* de máquinas virtuais, com destaque para a coleta das três informações que serão manipulados pela rede neural para classificação dos estados de operação do *cluster*: tempo de resposta, consumo de energia e carga de requisições disparadas para os servidores virtuais *Web*.

A seqüência de ações da política de reconfiguração original foi elaborada a partir da execução de testes de processamento junto ao *cluster* de servidores virtuais. Durante essa bateria de testes foram observados os impactos de cada ação de reconfiguração do ambiente, e verificados quais os valores limites de cada métrica utilizada (utilização de CPU, freqüência de CPU, etc.) que disparam intervenções para reconfiguração do *cluster,* conforme roteiro de ações descrito acima. Assim, os indicadores utilizados e as operações realizadas no ambiente foram ajustados de acordo com o melhor comportamento do *cluster* perante os valores de tempo de resposta medidos, de forma que a intervenção escolhida para determinada etapa de reconfiguração atue efetivamente para garantir o cumprimento das restrições de operação do ambiente.

Entretanto, apesar da política de reconfiguração ser baseada em observações do próprio ambiente de processamento, ela não leva em consideração oscilações em outros indicadores de desempenho além do tempo de resposta e não analisa a situação corrente de carga submetida ao *cluster* para o seu processo de tomada de decisão. Esta situação pode acarretar em uma baixa eficiência no gerenciamento de recursos do ambiente. Por exemplo, caso a quantidade de requisições disparadas contra o conjunto de servidores virtuais suba rapidamente para valores próximos ao limite suportado, a arquitetura realiza paulatinamente todo o processo de reconfiguração para adequar o *cluster* a este novo cenário de carga, pois a seqüência de operações não é flexível. Este cenário foi implementado na avaliação de desempenho em [7], sendo possível observar que o tempo de resposta do ambiente ficou acima do limite pré-estabelecido por um determinado período de tempo até que a arquitetura conseguisse configurar os recursos utilizados pelos servidores virtuais para adequá-los ao cenário observado.

Como a arquitetura original não considera em seu processo decisório a potência elétrica consumida pelo conjunto de servidores físicos, algumas ações de retirada de recursos podem se tornar inócuos perante este indicador. Em um cenário de transição de carga média ou alta para baixa, as retirada de recursos do ambiente para economia do consumo de energia poderia ser feita de forma mais direta, pois o impacto no tempo de resposta não será muito relevante em virtude da perspectiva de queda na quantidade de requisições submetidas ao *cluster*. Desta forma, seria possível manter o tempo de resposta abaixo do valor máximo estabelecido, e prover uma redução mais significativa do consumo de energia do ambiente.

Para efetuar um tratamento consistente das situações exemplo elencadas acima e realizar um mapeamento mais completo dos estados de operação do *cluster*, considerando suas transições e oscilações decorrentes do dinamismo dos cenários de carga submetidos, é proposta a nova política de reconfiguração que se encontra descrita na Seção 3.4. Assim como na proposta original, a nova política de reconfiguração também é derivada de testes de processamento junto ao ambiente. Para realizar a análise, tratamento e classificação dos dados obtidos pela medição do tempo de resposta, carga submetida e consumo de energia, a rede neural de mapas auto-organizáveis será inserida na arquitetura. Com o mapeamento dos estados de operação é possível tratar adequadamente transições de um estado A para outro estado B, indicando qual intervenção no ambiente é mais apropriada para a referida transição.

3.3 Validação da rede neural proposta

Conforme apontado na seção anterior, o processo de tomada decisão da nova política de reconfiguração é subsidiado por uma rede neural artificial com utilização de mapas auto-organizáveis. Assim, faz-se necessário em primeiro lugar verificar se esta rede neural será capaz de identificar os padrões de comportamento do *cluster* de acordo com o cenário de operação presente no escopo deste estudo. Este procedimento se justifica pelo histórico de insucessos observados vários casos descritos na literatura, onde a rede neural utilizada não é capaz de ser configurada e processada para situações diversas, seja por falhas no mapeamento do processo onde a mesma seria utilizada ou por limitações impostas pelas próprias características da rede neural.

Para que a rede neural utilizada neste estudo pudesse ser configurada e ajustada, são feitos testes preliminares com a arquitetura original. Durante esta etapa, as medições efetuadas junto ao ambiente foram filtradas e posteriormente submetidas à rede neural de forma a identificar se a mesma é capaz de reconhecer corretamente os padrões de comportamento do *cluster* de máquinas virtuais. Desta forma, os testes iniciais têm por objetivo configurar os parâmetros utilizados na rede neural e indicar que a presente proposta é viável, possibilitando assim a execução de testes mais completos para validar a solução implementada neste trabalho.

A infra-estrutura utilizada para suportar o ambiente de teste é indicada na Figura 11 e possui 6 servidores Linux, um switch Gigabit Ethernet e um *nobreak* com capacidade de 1500VA e fator de potência aproximado de 0.6. Cabe ressaltar que apenas os servidores físicos hospedeiros, cujas condições de operação são alvo deste estudo, são alimentados pelo *nobreak*. O referido equipamento fornece através de uma interface USB informações sobre a potência de energia entregue aos equipamentos a ele conectados, possibilitando assim a coleta periódica do consumo de energia dos servidores hospedeiros.

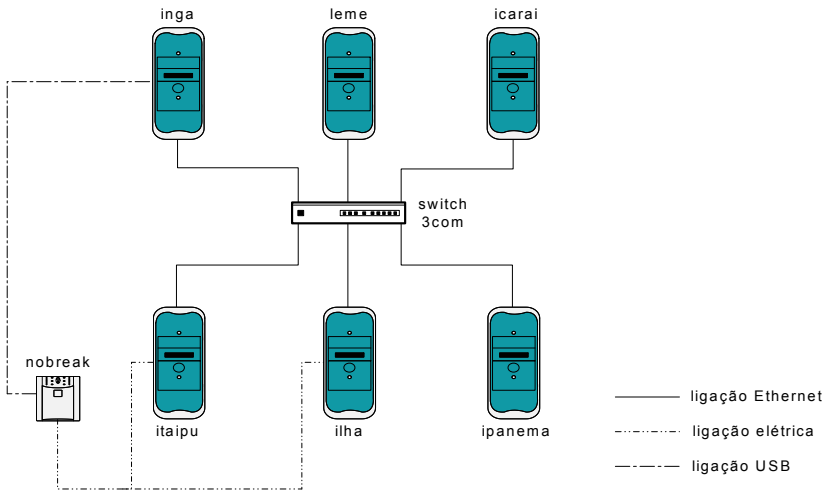


Figura 11 – Infra-estrutura utilizada

O conjunto de servidores Linux que formam a infra-estrutura é constituído de elementos heterogêneos, onde cada um possui um papel específico de operação, conforme indicado na tabela 12. O objetivo deste conjunto de equipamentos é simular o funcionamento de um *cluster* de servidores *Web* sendo submetidos a requisições HTTP geradas por clientes e distribuídas por um balanceador. São medidos o tempo de resposta e a carga de requisições, ambos coletados no servidor balanceador, e o consumo de energia. Conforme já mencionado anteriormente, essas três medidas são a entrada de dados da rede neural.

Tabela 12 – Configuração dos servidores físicos utilizados

Nome	Papel	Configuração	SO
Inga	Gerente/Monitor	Pentium 4 2.40 GHz 1 GB RAM e 40 GB HD	Ubuntu 8.04 Krn 2.6.24
Icarai	Cliente	Pentium 4 3.20 GHz 2,5 GB RAM e 80 GB HD	Ubuntu 8.04 Krn 2.6.24
Ipanema	Balanceador	Pentium D 3.00 GHz 4 GB RAM e 80 GB HD	Ubuntu 8.04 Krn 2.6.24
Itaipu	Hospedeiro Principal	Core2 Duo 3.00 GHz 4 GB RAM e 160 GB HD	Ubuntu 8.04 Krn 2.6.24-xen
Ilha	Hospedeiro Secundário	Core 2 Duo 3.00 GHz 4 GB RAM e 160 GB HD	Ubuntu 8.04 Krn 2.6.24-xen
Leme	Monitor CACTI	Pentium D 3.00 GHz 2 GB RAM e 80 GB HD	Fedora 5 Krn 2.6.20

O gerenciador Inga executa o sistema de gerência e um monitor de informações de desempenho. O cliente Icarai simula solicitações de clientes *Web* utilizando o gerador de carga HTTPERF [32] e a ferramenta Autobench [33], criando, assim, requisições HTTP com diversas taxas. O balanceador Grajaú, por sua vez, é o alvo das requisições geradas e executa

o servidor *Web* Apache com o módulo *mod_proxy_balancer* para distribuí-las entre os servidores *Web* em execução nas VMs. Os hospedeiros Itaipu e Ilha executam o virtualizador Xen para hospedar as VMs. O monitor Leme executa a ferramenta CACTI [34] para coletar e exibir em tempo real as informações de desempenho de todos os equipamentos. O *no-break* APC é conectado via USB ao gerenciador Ingá, já que este servidor executa as funções de gerenciador e monitor.

Em observação às restrições de operação já descritas em [7] do ambiente implantado, onde o mesmo suporta uma carga máxima de até 115 requisições por segundo, o cenário escolhido para esta etapa de testes é o *workload* em rampa conforme. Desta forma, a taxa de requisições por segundo inicia de forma crescente, começando com o valor igual a 5 e atingindo o valor limite de 115 após 60 minutos, efetuando assim um incremento de 10 requisições a cada 5 minutos. Após a marca de 60 minutos, a taxa de requisição passa a ser decrementada na mesma razão finalizando o experimento no tempo total de 120 minutos. O comportamento deste *workload* está ilustrado na Figura 13.

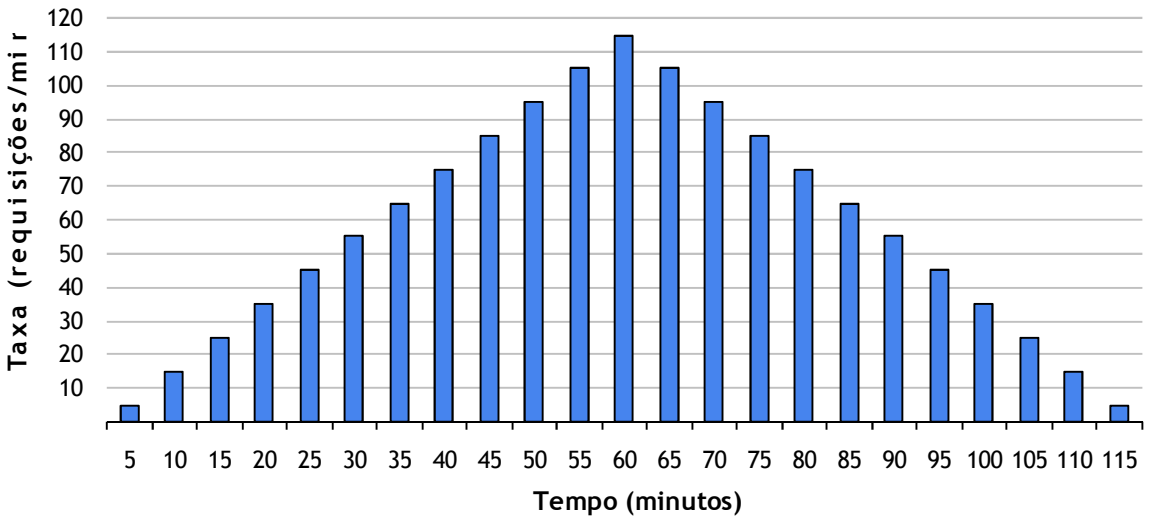


Figura 13 – Taxa de requisições submetidas ao *cluster* no *workload* em rampa

A coleta dos valores medidos de tempo de resposta, consumo de energia e carga submetida é efetuada a cada 15 segundos. Este intervalo entre as medições acarreta em 20 medições distintas em cada degrau da rampa, totalizando 460 medidas durante os 115 minutos de execução. O *workload* em rampa é executado 5 vezes, com repetição dos parâmetros utilizados de forma a permitir o cálculo da média e do desvio padrão dos valores medidos. Para eliminar eventuais desvios de funcionamento do *cluster*, os valores apresentados para treinamento da rede neural são os valores médios observados nas 5 execuções. Cabe salientar

que a execução do *workload* é realizada junto à arquitetura com base na política de reconfiguração original descrita na seção 3.2. Assim, para esta bateria de testes não são feitas as alterações na política de reconfiguração propostas neste trabalho, pois o objetivo desta etapa é observar o comportamento da rede neural para os valores de entrada a ela submetidos.

A rede neural construída para os testes preliminares utiliza a técnica de mapas auto-organizável, e é constituída de três neurônios na cama de entrada e cem neurônios na camada escondida. A Figura 14 indica a arquitetura da rede neural construída, com cada neurônio da camada de entrada conectado a cada um dos cem neurônios na camada escondida. Conforme observado no Capítulo 2, os neurônios da camada de entrada devem representar os eventos a serem tradados pela rede neural, que neste estudo são tempo de resposta, consumo de energia e carga submetida.

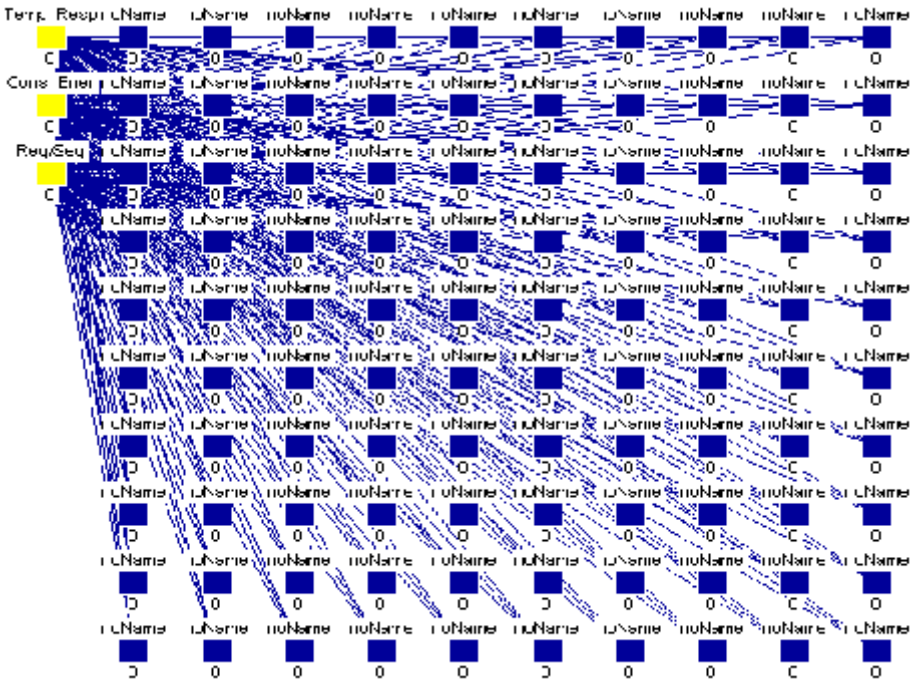


Figura 14 - Arquitetura da rede neural construída neste trabalho

Para o treinamento da rede neural descrita acima são utilizados os parâmetros descritos na Tabela 15. Em virtude da necessidade de normalização dos valores utilizados pela rede neural, as informações apresentadas na camada de entrada foram normalizadas através da Equação 7. Assim, os valores normalizados das medições obtidas do ambiente de servidores virtuais caracterizaram a etapa de treinamento. Para validação e observação do comportamento da rede neural são submetidos à camada de entrada os mesmos valores utilizados na fase de treinamento, utilizando uma ordem de apresentação aleatória.

Tabela 15 – Parâmetros iniciais definidos para a rede neural

Parâmetro	Valor
Peso de Adaptação *h(0)*	0.9
Raio de Adaptação *r(0)*	5
Taxa de decaimento do peso de adaptação *mult_H*	0.98
Taxa de decaimento do raio de adaptação *mult_R*	0.98
Tamanho horizontal da grade de neurônios	10
Iterações de Treinamento	10000

$$Norm = \frac{ValorCorrente - ValorMínimo}{ValorMáximo - ValorMínimo} \quad (7)$$

A Figura 16 mostra o resultado da formação de mapas topológicos formados através do agrupamento de estados semelhantes de operação do *cluster* de máquinas virtuais. A ferramenta SNNS possui um recurso de visualização de mapas chamado "Kohonen Maps". Esta funcionalidade transforma os resultados presentes no arquivo de saída apresentado, em um mapa topográfico bidimensional. Assim, é possível verificar a formação de quatro grupos de neurônios semelhantes, efetuando assim a classificação dos valores de entrada em quatro regiões distintas. É possível notar que essas quatro regiões encontram-se separados por fronteiras bem delimitadas, ou seja, nessas regiões de fronteira os neurônios não apresentam semelhanças significativas com seus vizinhos, o que indica um estágio de transição entre dois estados. Esta separação consistente das regiões de classificação indica que a rede neural identificou com sucesso os padrões de comportamento do *cluster* de acordo com as três variáveis analisada. Em virtude da carga de requisições submetidas ao ambiente de processamento ser controlada, obedecendo ao padrão de rampa, são identificados apenas quatro tipos de classificação de operação do *cluster*. Entretanto, com a utilização de uma carga real de submissões, que não obedece a padrões pré-estabelecidos, mais regiões de classificação são observadas no mapa, conforme é observado na seção 3.4 onde é utilizada como carga uma parcela dos *logs* de acesso ao site oficial da Copa do Mundo de 1998, que se encontra disponível em [35].

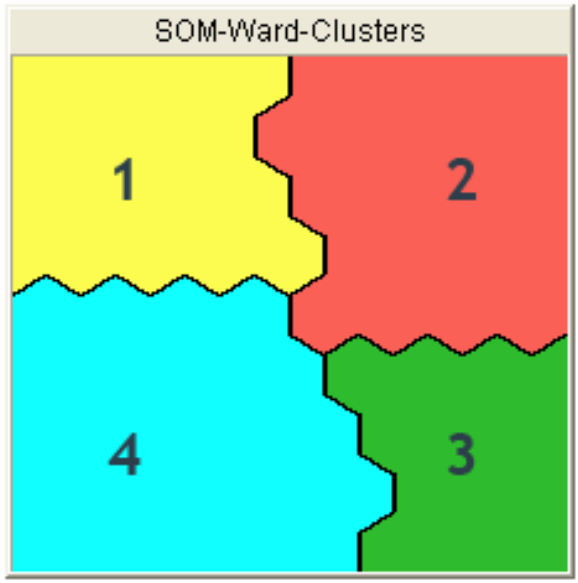


Figura 16 – Mapa topográfico gerado após a fase de treinamento da rede neural

3.4 Nova arquitetura e política de reconfiguração

Para implementação da nova política de reconfiguração, objeto deste trabalho, é adicionado à arquitetura original um módulo referente ao processamento da rede neural. Este módulo é incorporado ao módulo Controlador do sistema. Os demais componentes da arquitetura original não sofrem alterações, com exceção da API e suas seqüências de ações, que são adaptadas para implementar mais diretamente as intervenções no *cluster* indicadas pela rede neural, e encontram-se descritas nos Apêndices A e B, respectivamente. Para definição das ações de reconfiguração junto ao conjunto de servidores físicos, efetivadas através da API, é utilizado um cenário real de processamento de requisições obtido através de tráfego observado na Internet. Este cenário é baseado em *log*s de acesso ao portal oficial da Copa do Mundo de 1998 durante o período de realização do evento.

3.4.1 Configuração da rede neural

A configuração final de rede neural inserida na nova arquitetura da solução descrita neste trabalho é definida através do teste com *workload* variável, onde são realizadas alterações dos parâmetros de configuração da rede com o objetivo de melhorar a capacidade de identificação de padrões. A utilização do cenário de requisições variável se destina ao treinamento da rede neural para operação em um cenário genérico de requisições, pois o

workload escolhido apresenta diversas variações de carga, com a formação de vales e topos distintos. Desta forma, o treinamento da rede neural com o *workload* do site da Copa do Mundo de 1998, é capaz de definir a nova política de reconfiguração do ambiente de acordo com os estados de operação observados para o cenário de carga utilizado.

O período utilizado do *log* acima mencionado é compreendido entre às 12h do dia 04 de Julho de 1998 e às 23h59min do mesmo dia, totalizando assim 12 horas consecutivas de registros de requisições submetidas ao portal. Para atendimento às restrições de operação do *cluster*, que suportam até 115 requisições por minuto, conforme já observado na Seção 3.3, a quantidade de requisições foi normalizada para que o valor máximo de requisições, observado aproximadamente às 16h30min, seja equivalente a 115 requisições por minuto. O cenário normalizado de requisições pode ser observado na Figura 17.

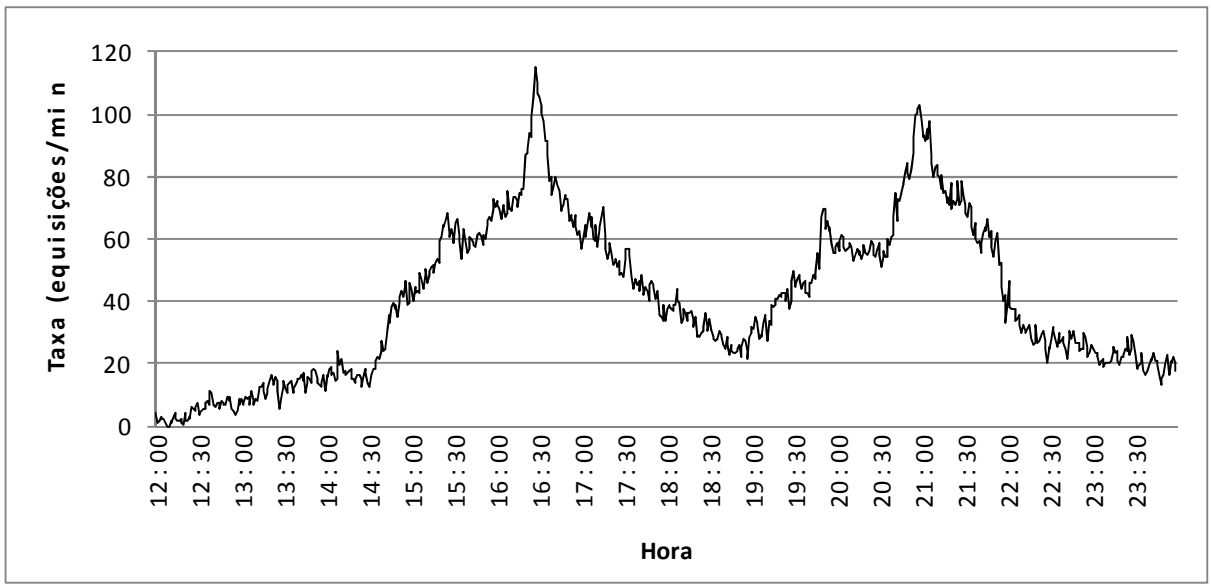


Figura 17 – Comportamento do *workload* da Copa do Mundo de 1998

Da mesma forma em que foram realizados os procedimentos para validação da rede neural com *workload* em rampa, descritos na seção 3.3, são realizadas cinco iterações de processamento do *workload* variável apresentado acima para treinamento da rede neural. Desta forma, são submetidos para processamento da rede neural os valores médios obtidos nos cinco ciclos de execução. A coleta das medidas referentes ao tempo de resposta, ao consumo de energia e à carga submetida, foi efetuada em períodos de 15 segundos, o que totaliza 2.880 registros para cada série de valores das três métricas utilizadas.

Cabe ressaltar que a utilização do *workload* variável eleva a complexidade dos padrões de dados submetidos para análise da rede neural, pois a carga em rampa apresenta um padrão linear de comportamento e a carga variável não obedece a situações pré-estabelecidas ou

controladas de requisições. Além disso, o quantitativo de séries de valores de entrada para a rede neural, que na fase de validação totalizaram 480, agora se apresenta em uma escala muito maior, cenário este que eleva a quantidade de ciclos de treinamento para além dos 10.000 de ciclos utilizados na fase de validação, por exemplo. Desta forma, os parâmetros de configuração da rede neural para esta etapa de treinamento diferem daqueles utilizados no processo de validação, e são apresentados na Tabela 18.

Tabela 18 – Parâmetros finais definidos para a rede neural

Parâmetro	Valor
Peso de Adaptação *h(0)*	0.85
Raio de Adaptação *r(0)*	5
Taxa de decaimento do peso de adaptação *mult_H*	0.90
Taxa de decaimento do raio de adaptação *mult_R*	0.98
Tamanho horizontal da grade de neurônios	10
Iterações de Treinamento	100000

É possível observar que são alterados os valores dos parâmetros (i) Peso de Adaptação, (ii) Taxa de decaimento do peso de adaptação e (iii) Iterações de Treinamento, em relação valores utilizados na fase de validação da rede neural. O Peso de Adaptação é reduzido de 0.90 para 0.85 para que a forma de adaptação dos neurônios vizinhos seja mais suave, em virtude do aumento da quantidade de iterações. O valor da Taxa de decaimento do peso de adaptação também é reduzido para 0.90, pois agora a duração do processo de treinamento será maior, e, caso a taxa original de 0.98 fosse utilizada, haveria uma diferença significativa entre os valores inicias e finais do peso de adaptação durante a etapa de treinamento, fato este que poderia gerar uma divergência no processo de formação das regiões classificadas pela rede neural.

Após a realização da execução do *workload* variável junto ao *cluster* de servidores virtuais durante o período de 12 horas, as 2.880 séries de registros são submetidas à camada de entrada da rede neural. Ao término da quantidade de iterações descritas na Tabela 18, os resultados são armazenados no arquivo "classificação.net", que é analisado pela funcionalidade "Kohonen Map" da ferramenta SNNS. Através deste mapa topográfico gerado com base nas medidas de tempo de resposta, consumo de energia e requisições submetidas, obtidas junto ao conjunto de servidores físicos, é definida a nova política de reconfiguração que é descrita adiante na Seção 3.4.2.

O mapa topográfico gerado indica a presença de cinco regiões distintas de classificação, conforme pode ser observado na Figura 19. Cada região possui um neurônio denominado centróide, que é o neurônio que apresenta maior ativação perante o seu conjunto de vizinhos e é considerado o núcleo da região a qual está inserido. Assim, estão descritos a seguir os valores de referência do vetor formado em cada neurônio-núcleo das cinco regiões observadas:

Região 1: Tempo de Resposta = 32,63 ms; Potência Consumida = 155,24 W; Requisições = 41,86 req/min. Cor azul;

Região 2: Tempo de Resposta = 32,93 ms; Potência Consumida = 171,94 W; Requisições = 67,15 req/min. Cor vermelha.

Região 3: Tempo de Resposta = 33,67 ms; Potência Consumida = 215,58 W; Requisições = 106,73 req/min. Cor verde;

Região 4: Tempo de Resposta = 33,15 ms; Potência Consumida = 190,68 W; Requisições = 81,81 req/min. Cor amarela;

Região 5: Tempo de Resposta = 33,36 ms; Potência Consumida = 207,67 W; Requisições = 97,70 req/min. Cor roxa.

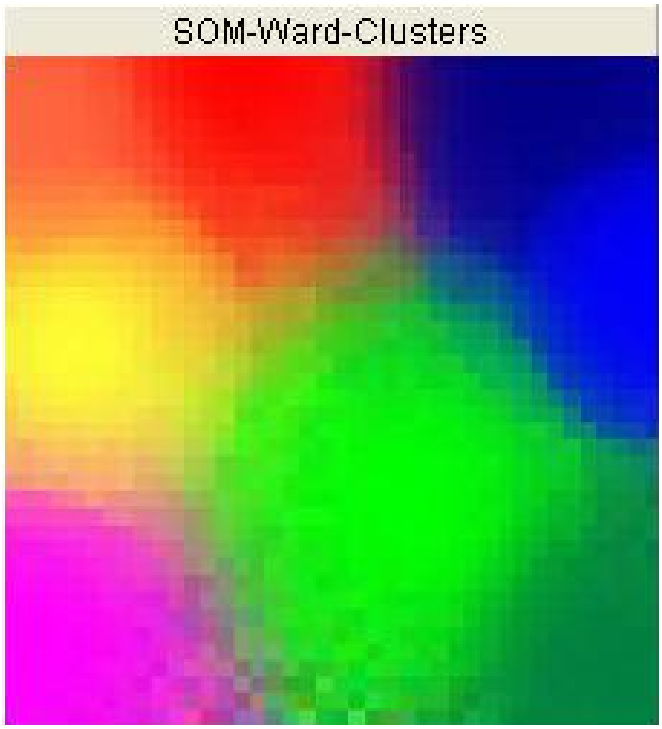


Figura 19 – Mapa topográfico final gerado após a fase de treinamento da rede neural

Para definição das ações faz-se necessário interpretar os valores dos centróides de cada região de acordo com o cenário de processamento utilizado para esta fase de treinamento da

rede neural. O objetivo é identificar e classificar um estado de operação do *cluster* baseado nos valores dos neurônios que compõem o núcleo de cada região. Após efetuar a classificação da região com base no neurônio centróide, o estado de operação classificado é válido para todo neurônio que estiver presente no interior da região. Isso significa que a malha de neurônios que está interna à região possui pesos sinápticos semelhantes. Esta semelhança é válida até área de fronteira da região selecionada, que é indicada pela mudança da cor referente às regiões, significando que os pesos sinápticos desses elementos presentes na área de fronteira não são semelhantes aos seus vizinhos, indicando que ali termina uma região e começa outra com características distintas.

Para facilitar a comparação entre os dados presentes em cada região, a Tabela 20 apresenta o percentual referente aos valores do núcleo de cada região em relação ao valor máximo de cada uma das três métricas verificado na fase de treinamento. Esta relação respeitou os aspectos de normalização das medidas observadas e é calculada através da Equação 7, descrita na Seção 3.3. Cabe ressaltar que para a medida de tempo de resposta é mantido o valor de referência de 33,5 ms representando o teto de valores permitidos para esta métrica. Esta adequação tem por objetivo evitar que o valor de 33,5 ms seja ultrapassado durante o funcionamento do sistema, pois podem ocorrer prejuízos para aplicação suportada caso o tempo de resposta esteja maior do que o limite pré-estabelecido. Para as métricas de potência consumida e requisições submetidas, os valores adotados foram os máximos medidos no teste com *workload* variável, 221,83 Watts e 115 req/min, respectivamente.

Tabela 20 – Valores percentuais de cada métrica para as regiões classificadas

	Temp. Resp.	Pot. Cons.	Req.
Região 1	33,3%	31,8%	22,2%
Região 2	56,3%	48,9%	49,1%
Região 3	113,8%	93,6%	91,2%
Região 4	73,5%	68,1%	64,7%
Região 5	89,3%	85,6%	81,6%

3.4.2 Descrição da nova política de reconfiguração

Definidas as regiões presentes no mapa topográfico resultante do processamento da rede neural, a nova política de reconfiguração dos servidores virtuais é especificada como um conjunto de possíveis transições entre as regiões do mapa. Assim, para cada região corrente

definida pela rede neural em um ciclo de monitoramento, é definida uma ação a ser solicitada à API, de acordo com a região vigente no ciclo de medição anterior. A escolha da melhor ação a ser tomada para cada região foi realizada através da realização de testes unitários, onde a carga de requisições é fixada para a taxa nominal indicada em cada região e são observados os impactos das ações possíveis perante o tempo de resposta e a potência consumida. É eleita a melhor intervenção aquela que efetua a redução mais significativa das duas métricas mencionadas, respeitando as restrições de operação do *cluster*. A descrição das funcionalidades implementadas pela API estão especificadas no Apêndice A, e as ações efetivadas pela API junto ao *cluster* de VMs de acordo com a região classificada são descritas no Apêndice B.

3.4.2.1 Região 1

A região 1 se caracteriza por baixas taxas em relação às três métricas verificadas pela rede neural em seu processamento. Este cenário possui uma quantidade de requisições pequena em relação aos demais momentos de processamento. Assim, esta situação de baixo número de requisições, além de tempo de resposta e consumo de energia também com valores reduzidos, permite uma ação mais agressiva de retirada de recursos do ambiente, com o objetivo de prover uma economia ainda maior de energia mantendo o tempo de resposta abaixo do limite de 33,5 ms. Assim, a ação de retirada de recursos do ambiente será analisada sob o viés de consolidação de todas as VM em um único servidor hospedeiro e intervenção nos núcleos de processadores, seja desativando núcleos ou reduzindo a freqüência de operação dos mesmos.

Para verificar qual a melhor configuração do servidor físico que consolida todos os servidores virtuais, os testes unitários contemplam as seguintes possibilidades: (i) dois núcleos de processadores ativos com freqüência máxima; (ii) um único núcleo de processador com freqüência máxima; e (iii) um único núcleo de processador com freqüência mínima. Como esta região apresenta os valores mínimos das três métricas observadas, não há necessidade de analisar possíveis transições de outras regiões para esta região 1. Conforme observado na Figura 20, a configuração (iii) é aquela que apresenta maior economia de energia além de não apresentar violação da restrição ao tempo de resposta máximo admitido. Assim, outras configurações possíveis para o hospedeiro que consolida as quatro VM em

operação são descartadas, pois a utilização de um único núcleo em sua freqüência mínima é a que melhor aos requisitos desejados pelo sistema.

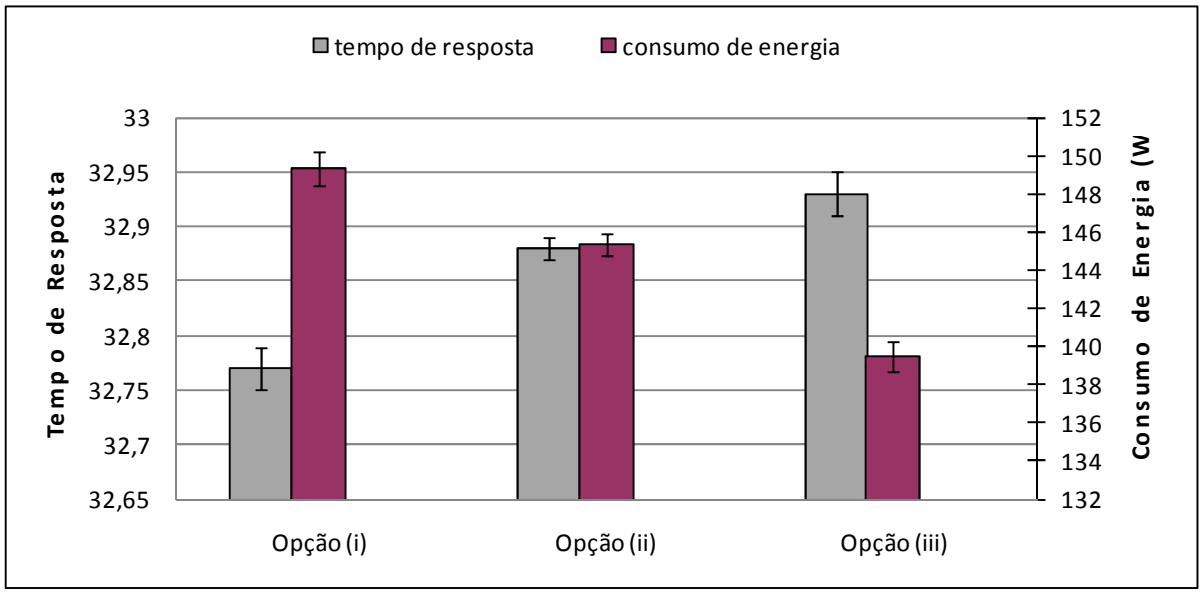


Figura 20 – Impactos das ações possíveis na região 1

3.4.2.2 Região 2

A região 2 indica que há uma carga mediana de requisições submetidas e que o tempo de resposta está em um valor acima da metade de seu valor máximo nominal. Entretanto, o consumo de energia permanece em um valor moderado, indicando que o sistema não está operando com a totalidade de seus recursos. Desta forma, a ação analisada para esta região visa adicionar recursos ao ambiente caso a região anterior seja a região 1, procedimento este que tem como objetivo evitar o aumento do tempo de resposta, que já se encontra com valor próximo ao máximo admitido. Caso os cenários anteriores sejam as regiões 3, 5 ou 4, há indicação de que o sistema está em uma transição de um período de alta carga de requisições submetidas para um período de carga mais moderada.

Assim, para uma transição entre as regiões 1 e 2, a ação associada consiste em ativar o servidor físico hospedeiro que se encontra fora de operação, dividir igualmente entre eles a quantidade de VMs suportadas, ou seja, cada servidor físico é responsável por dois servidores virtuais, e verificar qual a melhor configuração para os dois hospedeiros. Desta forma, são realizados testes unitários para três possibilidades de reconfiguração: (i) dois servidores hospedeiros com um único núcleo de processador cada em freqüência mínima; (ii) dois

servidores hospedeiros com um único núcleo de processador cada em freqüência máxima; e (iii) dois servidores hospedeiros com dois núcleos de processadores cada em freqüências mínimas.

A Figura 21 indica que a alternativa (i) é aquela que melhor atende a este cenário, pois é capaz de reduzir o tempo de resposta observado e não elevar demasiadamente o consumo de energia do ambiente. Cabe ressaltar que a opção (iii) é a mais eficiente na redução do tempo de resposta, mas eleva o consumo de energia de forma considerável. Como nesta região 2 o tempo de resposta ainda está dentro dos limites de operação, a opção (i) mostra-se mais interessante pois provê uma redução nesta métrica e não compromete o consumo de energia do ambiente.

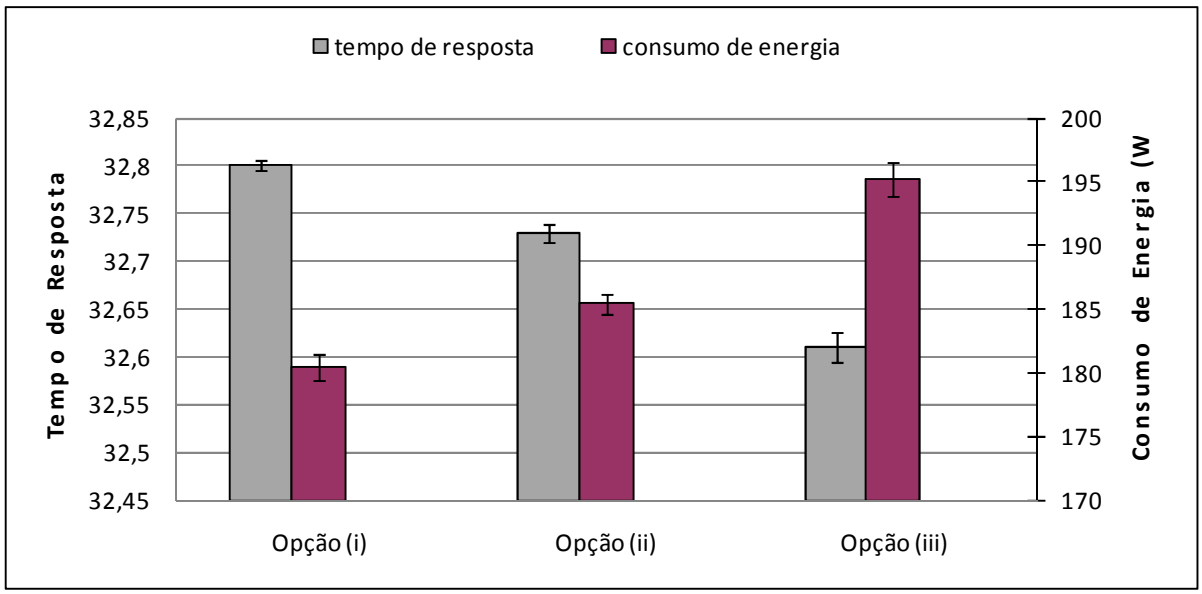


Figura 21 - Impactos das ações possíveis na região 2 para o cenário de carga crescente

Para o tratamento de uma transição entre as regiões 3, 5 ou 4 para a região 2, são analisadas alternativas de retirada de recursos do ambiente, pois a taxa de requisições e o tempo de resposta estão decrescentes. Assim, são verificadas as seguintes possibilidades de intervenção: (i) dois servidores hospedeiros com um único núcleo de processador cada em freqüência mínima; (ii) um único servidor hospedeiro com um único núcleo de processador em freqüência máxima; e (iii) um único servidor hospedeiro com dois núcleos de processador ativos em freqüência máxima. Conforme observado na Figura 22, a opção (iii) é aquela que apresenta os melhores resultados, pois a alternativa (ii) acarreta em uma violação da restrição de operação do tempo de resposta de acordo com o intervalo de confiança utilizado, e a opção

(i) consegue manter o tempo de resposta dentro do limite estabelecido, mas apresenta um consumo de energia maior do que a intervenção (iii).

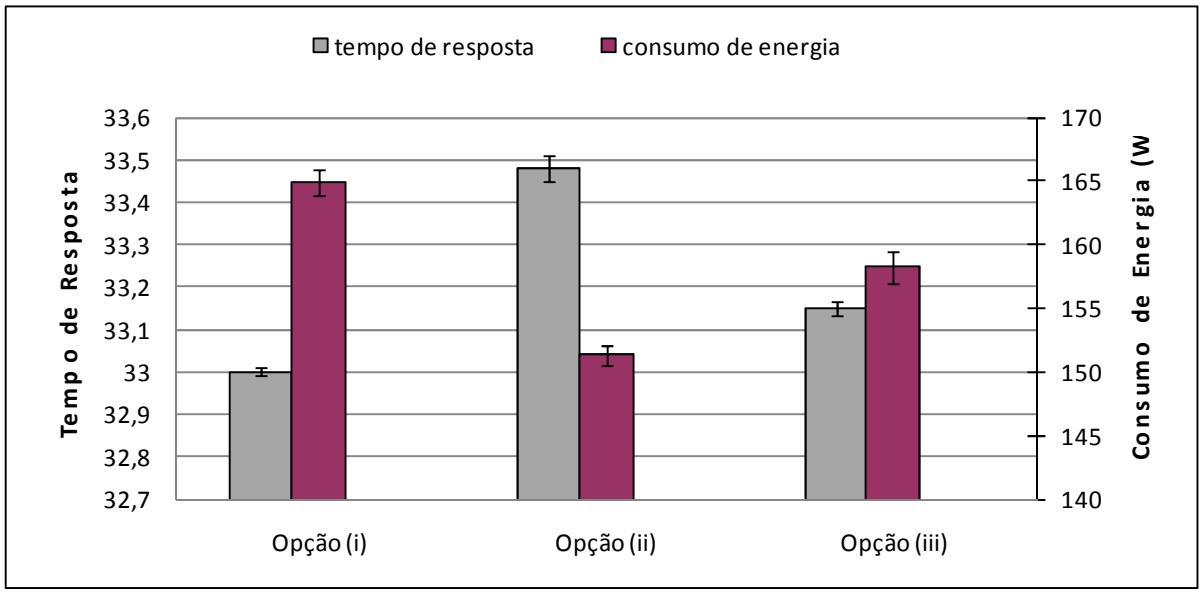


Figura 22 - Impactos das ações possíveis na região 2 para o cenário de carga decrescente

3.4.2.3 Região 3

A região 3 é caracterizada por uma clara violação do tempo de resposta da aplicação, que apresenta um valor maior que os 33,5 ms estipulado como o limite para a operação do sistema. Além do tempo de resposta estar elevado, a taxa de requisições submetidas ao *cluster* também é elevada, o que constitui em um cenário crítico com violação do tempo de resposta e carga elevada junto às VMs. Desta forma, as ações estudadas para este estudo estão relacionadas à disponibilização de recursos ao ambiente, ativando todos os núcleos em suas freqüências máximas, e balanceando os servidores virtuais de forma equilibrada entre os dois hospedeiros físicos, ou seja, cada servidor físico passa a suportar dois servidores virtuais.

Como se trata de uma região crítica, com violação das restrições de operação do sistema, a configuração analisada consiste em ativar todos os hospedeiros, balancear as VM entre eles e ativar os dois núcleos de cada servidor físico, independente de qual seja a região anterior. Esta configuração com disponibilidade total de recursos ao ambiente é capaz de trazer o tempo de resposta para um valor abaixo do limite pré-estabelecido, registrando a medida de 33,32 ms com um intervalo de confiança de 0,11. É importante ressaltar que com todos os recursos disponíveis do ambiente em operação, o consumo de energia medido após a

nova configuração torna-se maior do que o valor nominal especificado na região 3, assumindo o valor medido de 219,23 Watts com intervalo de confiança de 1,25. Entretanto, o objetivo da intervenção é regularizar o tempo de resposta verificado, mesmo que para isso seja necessário um consumo de energia ligeiramente maior.

3.4.2.4 Região 4

A região 4 indica que não há restrição de operação para a métrica de tempo de resposta, a taxa de requisições submetidas se apresenta em um valor moderado, enquanto o consumo de energia apresenta uma valor ligeiramente elevado. Assim, a ação de reconfiguração a ser realizada para a transição das regiões 3 ou 5 para a região 4 tem por objetivo reduzir o consumo de energia do ambiente, atentando para o fato de que o número corrente de requisições submetidas estar em um valor intermediário entre a metade e o valor máximo suportado pelo *cluster,* e uma abordagem muito agressiva de retirada de recursos do ambiente pode acarretar em piora do tempo de resposta, com risco de violação do valor limite de 33,5 ms. Por outro lado, uma transição oriunda das regiões 1 ou 2 significa que a carga submetida ao cluster está crescente e devem ser disponibilizados recursos ao ambiente para evitar uma possível violação das restrições de operação, pois o valor do tempo de resposta começa a se aproximar de seu valor limite.

Para reduzir o consumo de energia do ambiente, atividade que será realizada na transição das regiões 3 ou 5 para a presente região 4, os cenários avaliados nesta região devem retirar recursos disponíveis para o funcionamento do *cluster*. Cabe salientar que esta região possui os dois servidores físicos em operação, em virtude do mapeamento descrito na Seção 3.3.1. Desta forma, são analisadas três alternativas de retirada de recursos para escolha da intervenção associada a esta região: (i) dois servidores hospedeiros com um único núcleo de processador cada em freqüência mínima; (ii) um único servidor físico com dois núcleos de processador ativos em suas freqüências máximas; e (iii) dois servidores hospedeiros com um único núcleo de processador cada em sua freqüência máxima. Conforme observado na Figura 23, a opção (i) deve ser a alternativa selecionada, pois a opção (ii) gera violação das restrições de operação para o tempo de resposta e a opção (iii) apresenta um consumo de energia maior que a alternativa escolhida, embora ambas sejam capazes de manter o tempo de resposta em valores abaixo do limite adotado neste estudo.

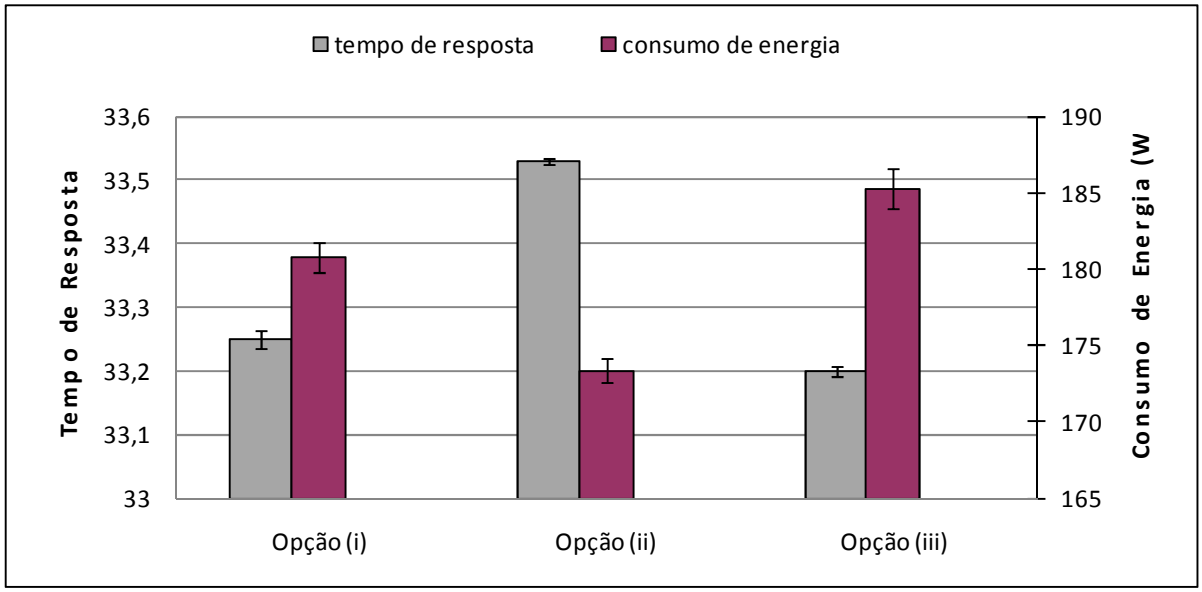


Figura 23 - Impactos das ações possíveis na região 4 para o cenário de carga decrescente

Para o cenário de crescente carga de requisições submetidas ao cluster de VM, representado pela transição das regiões 1 ou 2 para a região 4, não é imperativa a necessidade de acrescentar recursos ao ambiente, pois o tempo de resposta ainda se mantém dentro dos limites de operação na região 4. Entretanto, são avaliadas duas alternativas de entrega de recursos em ralação às configurações descritas nas regiões 1 e 2: (i) dois servidores hospedeiros com um único núcleo de processador cada em suas freqüências máximas, e (ii) dois servidores hospedeiros com dois núcleos de processador cada em suas freqüências mínimas. Esta avaliação tem por objetivo verificar principalmente o comportamento do consumo de energia, pois o tempo de resposta está aderente às restrições de operação, apesar de apresentar um valor próximo ao limite estabelecido. Assim, caso o consumo de energia não apresente uma elevação significativa após a reconfiguração para as alternativas (i) ou (ii), e o tempo de resposta apresente uma redução, é justificada a reconfiguração para este cenário de carga crescente.

Conforme observado na Figura 24, a opção (i) consegue efetuar uma redução do tempo de resposta observado e não elevou o consumo de energia verificado, já que o novo valor verificado após a reconfiguração está dentro do intervalo de confiança do valor original. A alternativa (ii) também é capaz de reduzir o tempo de resposta, mas esta reconfiguração eleva o valor do consumo de energia de forma considerável. Assim, a opção (i) é a escolhida

como reconfiguração a ser realizada na região 4 para um cenário de aumento das requisições submetidas ao *cluster* de servidores virtuais.

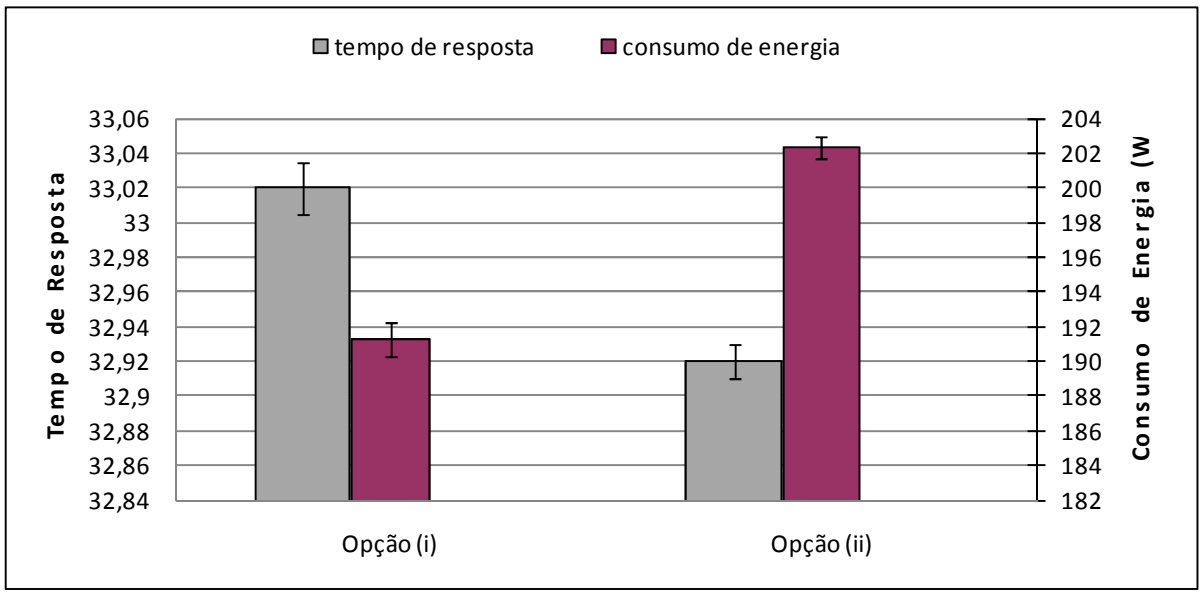


Figura 24 - Impactos das ações possíveis na região 4 para o cenário de carga crescente

3.4.2.5 Região 5

A região 5 apresenta valores elevados em todas as três métricas observadas, com destaque para o tempo de resposta que está próximo do limite estabelecido para a aplicação. Caso o cenário observado seja de transição entre a região 3, que apresentou violação das restrições de operação para o tempo de resposta, e a região 5, a retirada de recursos do ambiente deve ser realizada com cautela, pois mesmo com a taxa de requisições em queda o tempo de resposta observado na região 5 é alto. Desta forma, caso haja uma retirada de recursos significativa, o tempo de resposta pode voltar a apresentar valores acima do limite estipulado. Por outro lado, uma transição das regiões 1, 2 ou 4, que significam uma taxa crescente de requisições submetidas ao *cluster*, também é preocupante pois caso a quantidade de requisições continue apresentando um crescimento no decorrer do tempo, pode-se evoluir para o cenário da região 3.

Desta forma, para ambos os casos analisados a reconfiguração do ambiente presume disponibilidade quase total de recursos, de forma a evitar que o tempo de resposta permaneça nos níveis indicados pela região 5, reservando a entrega dos recursos em sua totalidade apenas quando o cluster de VM for classificado na região 3. As intervenções analisadas para a

presente região são: (i) dois servidores hospedeiros com um único núcleo de processador cada em suas freqüências máximas; e (ii) dois servidores hospedeiros com dois núcleos de processador cada em suas freqüências mínimas. A Figura 25 indica que a opção (ii) é a mais adequada para esta região, pois é capaz de efetuar uma redução do tempo de resposta, enquanto que a opção (i) não produz alteração na referida métrica, pois o novo valor verificado está dentro do intervalo de confiança do valor original.

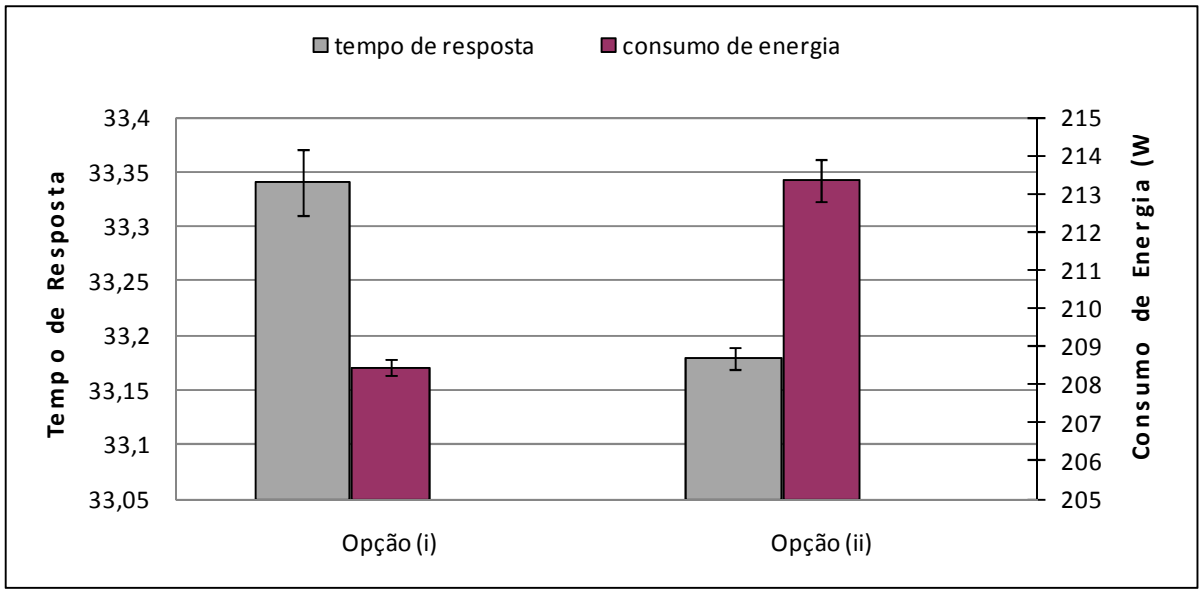


Figura 25 - Impactos das ações possíveis na região 5

3.4.3 Máquina de estados da nova política de reconfiguração

Para consolidar todas as ações descritas na seção anterior e identificar de forma mais clara as intervenções efetivas no cluster a cada transição de região, a Figura 26 descreve uma máquina de estados referente à nova política de reconfiguração do *cluster* de servidores virtuais. Cabe ressaltar que determinadas transições especificadas na política de reconfiguração não foram observadas durante a validação do treinamento da rede neural. Por exemplo, podemos citar a transição da região 1 para a região 5, que representa um crescimento abrupto da taxa de requisições e conseqüentemente do tempo de resposta observado. Entretanto, a nova política efetua o tratamento desta transição, e de todas as outras possíveis, pois a mesma pode ocorrer em um *workload* diferente daquele utilizado no treinamento da rede neural. Assim, como o objetivo de utilização da rede neural é a capacidade da mesma de responder corretamente a padrões diferentes daqueles utilizados na

fase de treinamento, faz-se necessário especificar todas as transições possíveis entre as regiões.

A máquina de estados referente à nova política de reconfiguração difere em vários aspectos da máquina de estados associada à política original. Dentre as principais mudanças, pode-se destacar:

- Para cada estado apresentado, há uma seqüência de ações efetivas pela API. Já na solução original cada determinado estado dispara uma única ação a ser concretizada pela API junto ao *cluster* de VMs;
- A transição entre os estados podem ser realizadas na proporção de N para N, evitando assim o escalonamento gradativo entre vários estados intermediários realizado pela política original;
- Independência entre as ações de cada estado perante os demais, pois as ações definidas para cada região classificada pela rede neural não possui vínculo com as ações anteriores efetivadas, e sim com a transição entre regiões.

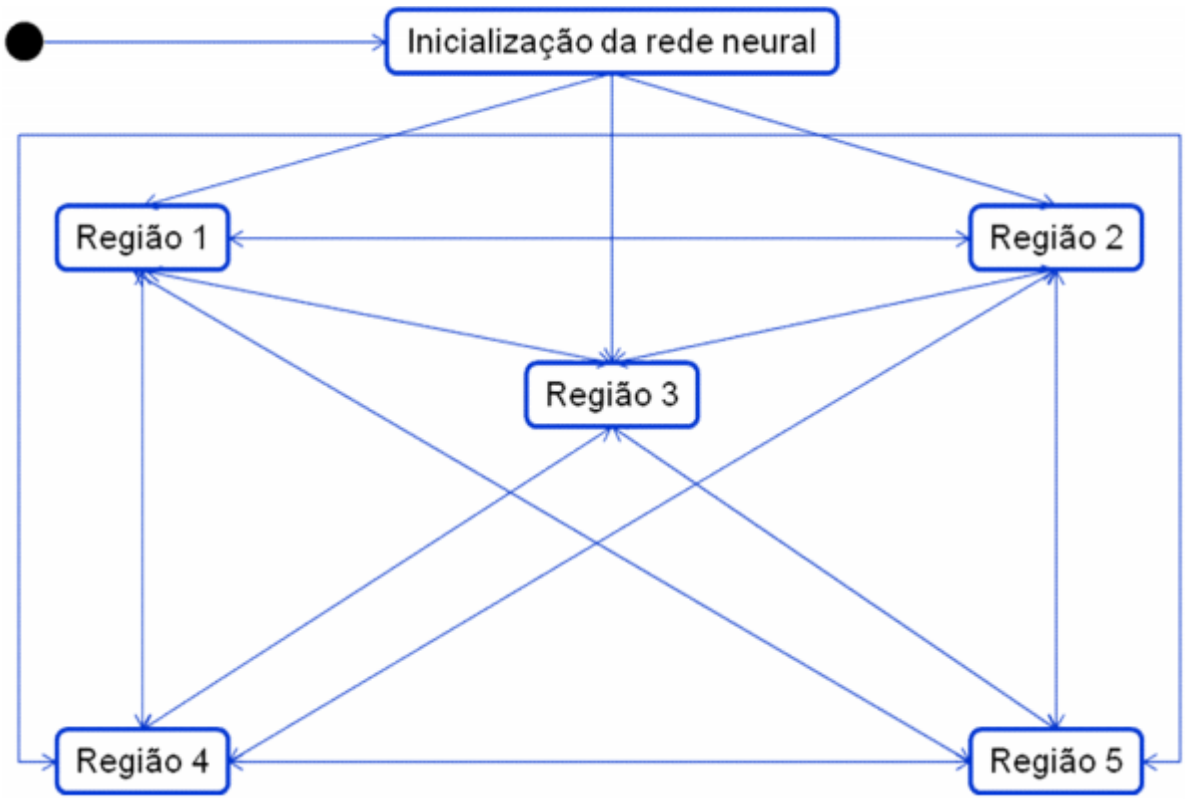


Figura 26 – Máquina de estados da nova política de reconfiguração

Cabe ressaltar que qualquer transição com origem na região 1 acarreta em ativação do hospedeiro 2, pois o mesmo encontra-se inativo através da intervenção definida para a região 1, e balancear de forma igualitária as VMs entre eles, ou seja, cada hospedeiro físico deve ser responsável em suportar 02 VMs. Por outro lado, transições com origem nas regiões 3, 4 ou 5 e destino para as regiões 1 ou 2 produzem a consolidação das VMs no hospedeiro 1 e inativação do hospedeiro 2. Estas intervenções efetivadas pela API junto ao conjunto de servidores virtuais são provenientes das ações definidas para cada região de classificação do estado de operação do *cluster*, conforme descrição da política de reconfiguração descrita na Seção 3.4.2.

CAPÍTULO 4 AVALIAÇÃO DE DESEMPENHO

Neste capítulo serão apresentados os testes realizados com a solução proposta neste trabalho. A avaliação de desempenho do sistema é realizada com três *workload*s distintos, onde os resultados verificados são comparados com aqueles observados através da utilização da arquitetura original. Os indicadores utilizados para comparação dos resultados entre as duas propostas são tempo de resposta e consumo de energia do conjunto de servidores físicos. Assim, a validação dos resultados deste estudo é efetuada com base na comparação do funcionamento da arquitetura original e da nova solução.

4.1 Configuração dos testes

Os testes realizados neste capítulo utilizaram a mesma infra-estrutura descrita na seção 3.3, onde foi efetuada a validação da rede neural inserida na nova arquitetura. Durante a execução dos testes foram coletadas informações sobre potência elétrica consumida e tempo de resposta da aplicação implementada. Cabe ressaltar que esses dois indicadores são utilizados como informação de entrada da rede neural, em conjunto com a carga de requisições corrente, para funcionamento da nova política de reconfiguração do ambiente.

Para que a rede neural possa efetuar a leitura dessas três métricas em sua camada de entrada foi desenvolvido um *script* para consolidação dos valores medidos durante a execução dos *workload*s. Esse *script* também realiza a formatação dos valores medidos em um layout específico de arquivo que será submetido como entrada para rede neural. Assim, com uma periodicidade de 5 minutos o *script* consolida as informações de tempo de resposta, potência consumida e carga de requisições obtidas pelo módulo *Coletor* e monta o arquivo de entrada para a rede neural, que irá efetuar o processamento dessas informações e efetuar a classificação do estado do *cluster* de acordo com os padrões de operação obtidos na etapa de treinamento descrita na seção 3.3.

A Figura 27 apresenta uma amostra do arquivo montado, que é sempre criado com o nome padrão *medidas.pat*. Como a coleta de informações do ambiente é realizada a cada 15 segundos, há sessenta registros de cada uma das três métricas coletadas do ambiente de processamento. Para cada uma das três métricas é calculada a média dos valores obtidos nas sessenta medições, e este valor médio é especificado em cada montagem do arquivo *medidas.pat*. Assim, os valores especificados em *"No. of patterns"* e *"No. of input units"* nas

linhas 5 e 6 são fixados em 1 e 3, respectivamente, em cada arquivo criado. Acima dessas especificações consta nas linhas 1 e 2 o cabeçalho definido pela ferramenta SNNS para arquivos de padrões de entrada. Sem este cabeçalho a ferramenta não é capaz de identificar que o arquivo de entrada, no caso o *medidas.pat,* é um arquivo válido para utilização da rede neural. Na linha 8 constam três colunas representando os valores médios das medidas de tempo de resposta, potência consumida e carga de requisições, nesta ordem. Como exemplo estão mencionados os valores nominais do centróide da região 1.

```
1    SNNS pattern definition file V3.2
2    generated at Thu Apr 14 19:56:37 2011
3
4
5    No. of patterns : 1
6    No. of input units : 3
7
8    32.63 155.24 41.86
```

Figura 27 – Exemplo do arquivo montado submetido para processamento da rede neural

Para uma avaliação mais completa do desempenho do sistema proposto neste estudo, são realizados testes com três tipos de *workloads* distintos. Os dois primeiros já foram apresentados neste trabalho e são o *workload* em rampa e o *log* de acesso ao portal da Copa do Mundo de 1998. O motivo da escolha desses *workloads* é efetuar uma comparação direta com os resultados obtidos pela arquitetura original que estão descritos em [7], pois a avaliação de desempenho do referido trabalho utilizou esses dois *workloads*. O terceiro cenário de carga escolhido foi o *log* de acesso à página da NASA em período do mês de Julho de 2005, também obtido em [35]. A razão da escolha deste terceiro *workload* foi verificar a capacidade de generalização do presente estudo, pois este novo cenário de carga escolhido difere por completo do *workload* da Copa de 1998 que foi utilizado na fase de treinamento da rede neural. Cabe ressaltar que a carga de requisições em rampa também apresenta um padrão diferente daquele apresentado na fase de treinamento da rede neural. Entretanto, este cenário não representa uma situação real de carga de requisições, pois o acréscimo e o decréscimo de requisições submetidas ao cluster são lineares e controlados.

A seguir são apresentados os resultados obtidos através da execução dos três *workloads* mencionados. Para cada um dos três cenários de carga são indicados os gráficos comparativos para as métricas de tempo de resposta e consumo de energia, que são os indicadores alvo que o sistema desenvolvido faz tratamento. Assim, os gráficos representam os comparativos entre os resultados obtidos pela arquitetura original e os resultados

verificados com a nova política de reconfiguração apresentada neste presente trabalho. A execução de cada *workload* foi replicada cinco vezes para permitir o cálculo da média e do desvio padrão dos valores medidos. Para apresentação dos resultados, foi utilizado um intervalo de confiança de 90% calculado com auxílio de uma distribuição *t de Student* com grau de liberdade igual a 4.

4.2 *Workload* em rampa

O comportamento da carga de requisições submetidas ao cluster de servidores virtuais no cenário em rampa encontra-se descrito na Figura 13 da Seção 3.2. Por se tratar de um incremento e um posterior decremento linear de requisições, o pico observado ocorre na metade do período do teste, ou seja, quando o teste está com 1 hora de duração. Neste momento, é atingido o limite suportado pelo *cluster* de 115 requisições por minuto. As Figuras 28 e 29 apresentam o comportamento do tempo de resposta da aplicação e do consumo de energia do ambiente, respectivamente, durante a janela de teste. São apresentados os comportamentos da referida métrica para a arquitetura original e com a utilização do sistema proposto neste trabalho.

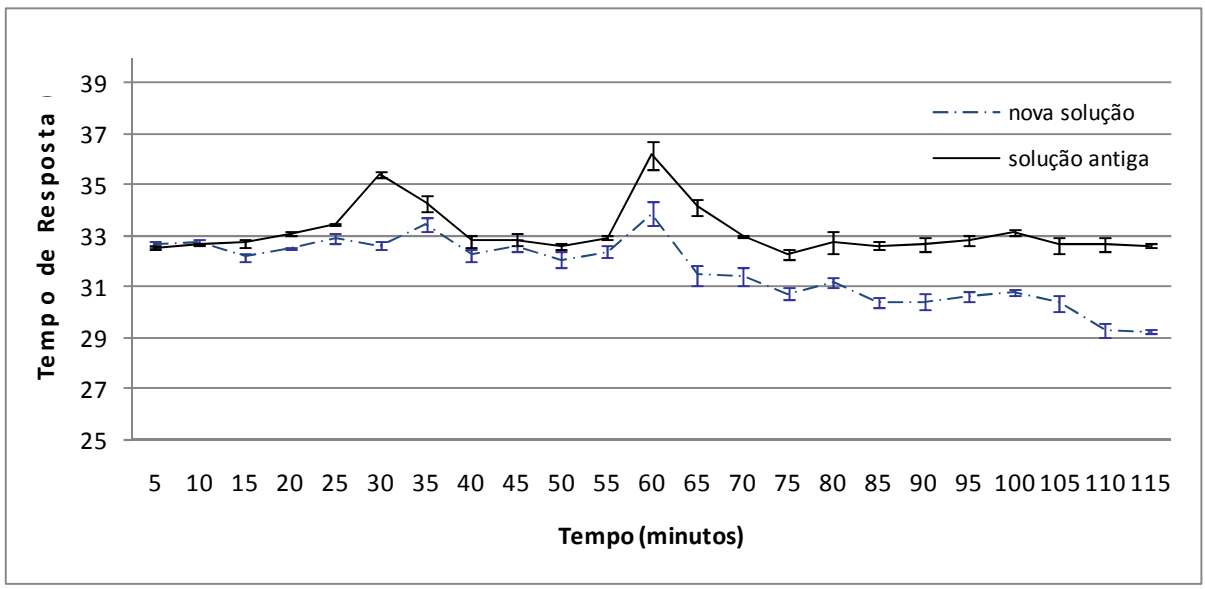


Figura 28 - Tempo de resposta para o *workload* em rampa

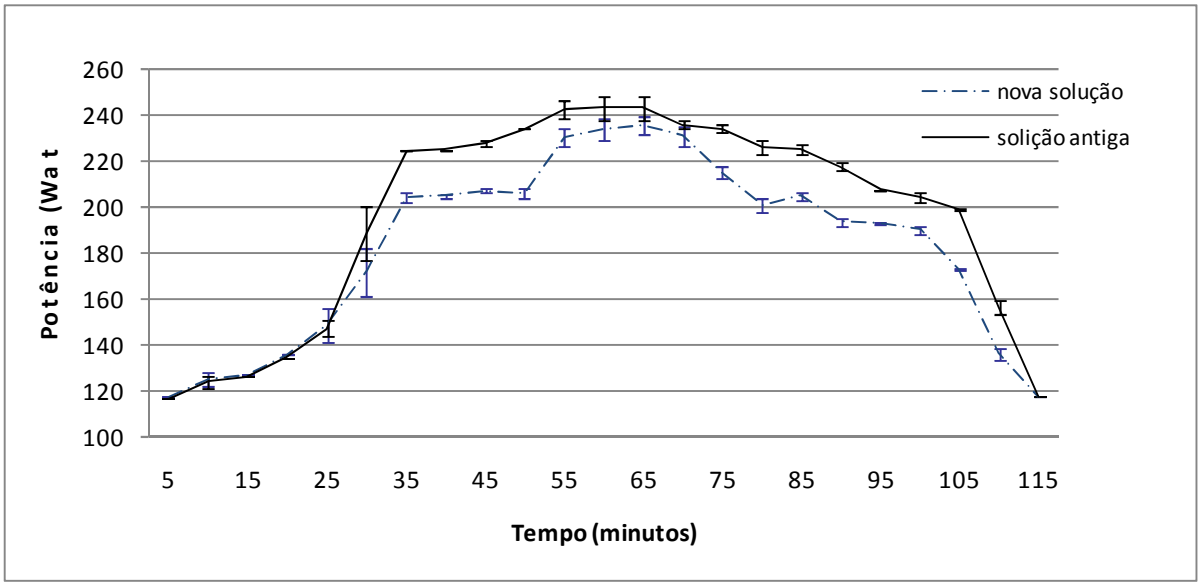


Figura 29 - Potência consumida para o *workload* em rampa

Durante as duas horas de execução dos testes são verificadas oito transições de estado de operação do *cluster* de acordo com as regiões classificadas na Seção 3.3. Desta forma, estas transições produzem oito intervenções de reconfiguração de recursos efetivadas pela API. A primeira delas ocorre aos 10 minutos, quando é verificada uma transição da região 1 para a região 2. A intervenção efetivada pela API por causa desta transição é responsável pela queda do tempo de resposta observado no instante igual a 10 minutos. A seguir, uma nova transição é observada aos 25 minutos, com origem na região 2 e finalização na região 4. Posteriormente, aos 35 minutos há transição da região 4 para a região 5. As três transições anteriores, que efetuaram consecutivas entregas de recursos ao ambiente são responsáveis pela curva ascendente do consumo de energia até a primeira metade do teste. Finalmente, aos 60 minutos há uma transição dentre as regiões 5 e 3, indicando que a classificação corrente é de uma região crítica, com possível violação da restrição de operação para o tempo de resposta. Neste momento, o *workload* em rampa está em seu pico de carga submetida ao *cluster*, e o sistema disponibiliza a totalidade de recursos possíveis, conforme a regra para intervenção definida para a região 3. Até este momento, todas as transições representaram um cenário de carga crescente, fato este que indica que o sistema está reagindo corretamente perante o *workload* utilizado.

É possível observar que a partir dos 10 minutos de teste os valores de tempo de resposta estão abaixo daqueles observados pela solução original. Já o consumo de energia

apresenta valores menores que a solução original entre os 40 e 55 minutos. Como aos 60 minutos é verificada uma transição para a região 3, oriunda da região 5, e todos os recursos computacionais estão ativos, o consumo de energia corrente verificado no sistema se iguala ao observado na solução original, de acordo com o intervalo de confiança utilizado. Esta igualdade permanece até o instante igual a 70 minutos, quando o já se encontra em sua fase decrescente e é efetuada uma transição da região 3 para 5, realizando a primeira retirada de recursos do ambiente deste teste. Pode-se notar uma queda significativa do consumo de energia entre os 70 e 75 minutos, redução esta viabilizada através da retirada de recursos do ambiente, e uma pequena elevação do tempo de resposta. Aos 80 minutos uma nova transição ocorre, desta vez da região 5 para 4. Há um comportamento semelhante em *workload* relação à região anterior, com consumo de energia decrescente e uma pequena oscilação para cima do tempo de resposta, mas que em linhas gerais também apresenta uma curva decrescente em virtude da quantidade de requisições submetidas também estar caindo.

Por fim, duas últimas transições são observadas aos 100 e 110 minutos, das regiões 4 para 2 e 2 para 1, respectivamente. Entretanto, como a carga de requisições nesses instantes já é relativamente baixa, não há alterações consideráveis no tempo de resposta e consumo de energia, e ambos continuam a apresentar curvas descendentes em seus respectivos gráficos. Cabe ressaltar que após a metade do teste, ou seja, após o pico de requisições e o início da fase decrescente de carga, todas as transições realizadas mapearam corretamente esta fase de redução de requisições submetidas e efetuaram intervenções para retirada de recursos do ambiente, propiciando assim valores de consumo de energia menores do que os medidos durante o mesmo teste com a arquitetura original.

Em relação à arquitetura original a presente proposta apresentou uma redução global de consumo de energia de 15,3% durante as duas horas de realização do teste. O melhor desempenho ocorreu entre os instantes de 35 e 60 minutos e 75 e 90 minutos, com uma redução média de 19,4% e 17,2%, respectivamente. Essas reduções observadas são justificadas pelo fato de que nas duas janelas de mencionadas o sistema não disponibiliza todos os recursos disponíveis para utilização do *cluster*, conseguindo prover um consumo de energia menor que a solução original. Quando o estado de operação é classificado na região 3 entre 60 e 70 minutos, não é possível observar redução do consumo de energia de acordo com o intervalo de confiança utilizado, pois tanto o sistema original quanto a nova solução proposta operam com a totalidade de recursos computacionais disponíveis. Em relação ao tempo de resposta, a utilização da nova solução acarretou violação das restrições de operação

apenas no instante igual a 65 minutos, de acordo com o intervalo de confiança utilizado, e viabiliza em redução global de 21,4% perante a arquitetura original.

4.3 *Workload* com *logs* da Copa do Mundo de 1998

A realização do teste com o *workload* extraído dos *logs* de acesso ao portal da Copa do Mundo de 1998 tem por finalidade observar o comportamento do sistema perante uma carga de requisições variável, sem padrão de comportamento previamente definido. Apesar deste cenário de carga ser o que efetuou o treinamento da rede neural, a realização de seu teste é válida para efetuar um comparativo entre a solução original e a presente solução, pois este foi o principal *workload* utilizado na avaliação de desempenho da arquitetura original. A realização desta bateria de teste teve a duração de doze horas, em contraste às duas horas de duração do teste com o *workload* em rampa. Este período estendido de duração do teste ocorre pela necessidade de se observar o comportamento do sistema de uma forma mais completa, verificando a transição de estados entre as classificações do cluster durante as oscilações de carga observadas neste novo cenário de carga.

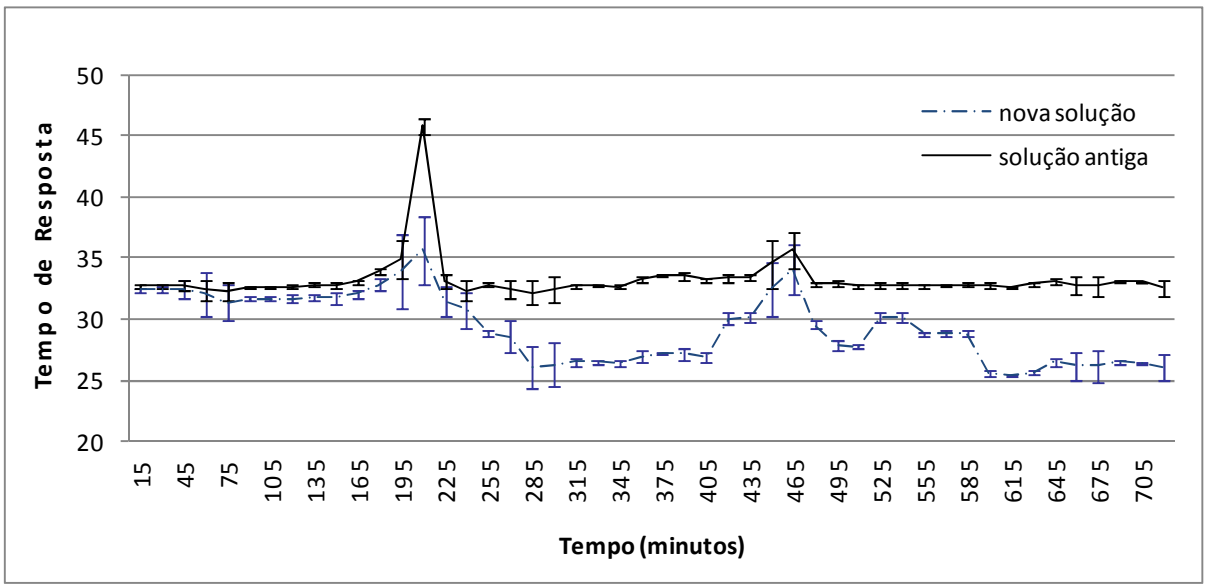


Figura 30 – Tempo de Resposta para o *workload* da Copa de 98

A Figura 15 apresentada na Seção 3.3.1 o comportamento da carga de requisições utilizada neste teste. A Figura 30 apresenta o comportamento do tempo de resposta durante a realização

do teste a Figura 31 indica a variação do consumo de energia medido do ambiente. Como este *workload* foi o mesmo padrão utilizado para o treinamento da rede neural, espera-se que os resultados obtidos neste teste apresentem taxas de desempenho melhores do que aquela observada no *workload* em rampa. Esta expectativa se baseia no fato de que os padrões submetidos à rede neural são os mesmos apresentados na fase de treinamento, salvo eventuais desvios decorrentes do processo de medição.

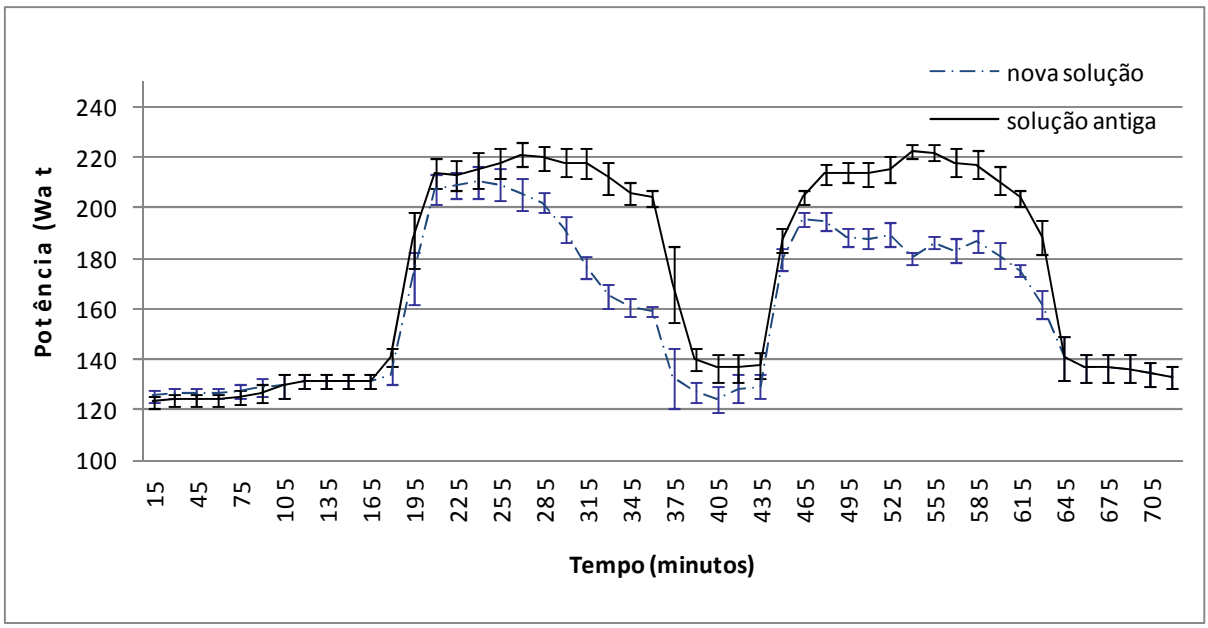


Figura 31 – Consumo de energia para o *workload* da Copa de 98

No decorrer da execução do teste são observadas doze transições entre as regiões classificadas pela rede neural. A primeira transição é observada aos 60 minutos, quando a classificação anterior referente à região 1 é atualizada para a região 2. A próxima transição é realizada aos 180 minutos, da região 2 para a região 4. Nesta transição é possível observar que a mesma não é capaz de reduzir o tempo de resposta, que se mantém estável de acordo com o intervalo de confiança utilizado, e gera uma elevação considerável do consumo de energia do ambiente. No instante igual a 195 minutos de duração do teste é feita uma transição da região 4 diretamente para a região 3, sem passar pela região 5. Isto ocorre em virtude do tempo de reposta apresentar comportamento crescente, ocorrendo inclusive violação da restrição de operação de acordo com o intervalo de confiança, que seu limite superior apresenta o valor de 33,90 ms. Com a intervenção no cluster realizada na região 3, como todos os recursos

disponibilizados ao *cluster*, nota-se que o tempo de resposta passa a ser decrescente a partir dos 200 minutos de teste. Entretanto, o consumo de energia assume seus valores máximos observados durante esta bateria de teste, e atinge valores semelhantes, com intervalos de confiança sobrepostos entre as duas curvas do gráfico, àqueles observados durante a execução com a arquitetura original.

Este cenário de consumo de energia em seu valor máximo permanece até os 270 minutos quando há uma nova classificação para a região 5. Assim, uma pequena fatia de recursos é retirada do cluster e o consumo de energia começa a cair, ainda que de forma moderada. Somente aos 285 minutos, quando há uma transição da região 5 para a região 4, é que se observa uma queda acentuada no consumo de energia medido do ambiente. Como é realizada uma nova retirada de recursos do ambiente, procedimento este responsável pela queda do consumo de energia mencionada, o tempo de resposta que estava com valores decrescentes passa a ter comportamento estável. Aos 345 minutos há uma nova transição, agora da região 4 para a região 2. Esta transição implica em mais uma queda do consumo de energia e não altera os valores de tempo de resposta em relação à classificação anterior, permanecendo os valores medidos dentro do intervalo de confiança utilizado. Finalmente, aos 375 minutos o *cluster* retorna ao seu estado original de classificação, quando ocorre uma transição da região 2 para a região 1. Neste momento, o consumo de energia continua decrescente rumo ao piso das medições, que ocorre aos 405 minutos quando o consumo de energia medido assume o valor de 124,48 W.

O cenário de consumo de energia reduzido é verificado até o instante igual a 420 minutos, quando tempo de resposta apresenta uma elevação, ocorrendo uma transição da região 1 para a região 2. A intervenção realizada no cluster é capaz de estabilizar o tempo de resposta e de efetuar apenas uma sutil elevação do consumo de energia. Entretanto, aos 440 minutos de testes um aumento mais significativo do tempo de resposta é verificado e uma transição da região 2 para a região 5 é realizada, sem passar pela região intermediária 4. Porém, a intervenção efetuada pela API na região 5 não é capaz de deter o aumento do tempo de resposta, que aos 450 minutos apresenta violação da restrição de operação, registrando um valor igual a 34,11 ms. A rede neural detecta esta situação, classificando o estado de operação do cluster na região 3. Neste instante, é possível observar um aumento acentuado do consumo de energias, que é fruto das seguidas intervenções que procederam entrega de recursos ao ambiente, em especial daquela realizada na região 3 que disponibiliza a totalidade de recursos para uso do *cluster*. Entretanto, esta intervenção é capaz de retornar o valor do tempo de

resposta para abaixo do valor limite estipulado, conforme pode ser observado pelo comportamento descendente da curva no gráfico a partir dos 465 minutos.

Aos 630 minutos é realizada uma transição da região 3 diretamente para região 2, sem necessidade de realizara transição entre as regiões intermediárias 5 e 4. Esta transição direta para a região 2 realiza um decremento acentuado do consumo de energia do ambiente. Finalmente aos 680 minutos é realizada a última transição, oriunda da região 2 para a região 1. Esta última transição não causa impacto significante no tempo de resposta que se manteve estável na parte final do teste, principalmente pela baixa carga de requisições submetidas ao *cluster* no período.

Em comparação com o desempenho da arquitetura original, a presente solução apresenta uma redução global média de 26,4% do consumo de energia durante a realização da bateria de teste. Os melhores resultados são verificados entre os períodos de 285 e 405 minutos, com uma redução média de 37,5% e entre os 495 minutos e 585 minutos, onde foi verificada uma redução média de 32,1%. Em relação ao tempo de resposta, no decorrer das 12 horas de teste foi observada uma redução média de 29,5%, com as melhores taxas verificadas entre os instantes de 255 a 405 minutos e 585 a 720 minutos, com redução média de 33,2% e 36,9%, respectivamente. Cabe ressaltar que ocorreu violação das restrições de operação do tempo de resposta em dois momentos: aos 180 e 450 minutos. Entretanto, é possível observar que os valores máximos obtidos nesses momentos de quebra da qualidade de serviço são menores do que aqueles observados pela utilização da arquitetura original.

4.4 *Workload* com *logs* de acesso da NASA

Para realização deste terceiro e último teste de desempenho da solução proposta neste estudo é utilizado o *log* de acesso á página da NASA (*National Aeronautics and Space Administration*, agência Americana para o Programa Aeroespacial) como cenário de carga. Este *workload* foi registrado entre as 12h e 23h59min do dia 17 de Julho de 2005. A quantidade de requisições verificadas originalmente no arquivo de *log* foi normalizada para que a quantidade máxima de requisições submetidas seja igual a taxa de 115 por minuto, respeitando assim os limites de operação do *cluster* de máquinas virtuais. A escolha deste *workload* para utilização nesta avaliação de desempenho se deve ao fato do comportamento quase que intermitente da quantidade de requisições ao longo do tempo. Assim, além de ter

um comportamento variável, sem regras de formação pré-estabelecidas, o padrão apresentado por este cenário de carga difere de forma considerável do *workload* da Copa do Mundo de 1998 utilizado na análise de desempenho da Seção 4.3. A Figura 32 apresenta o comportamento da carga de requisições submetidas ao *cluster* para a presente bateria de testes.

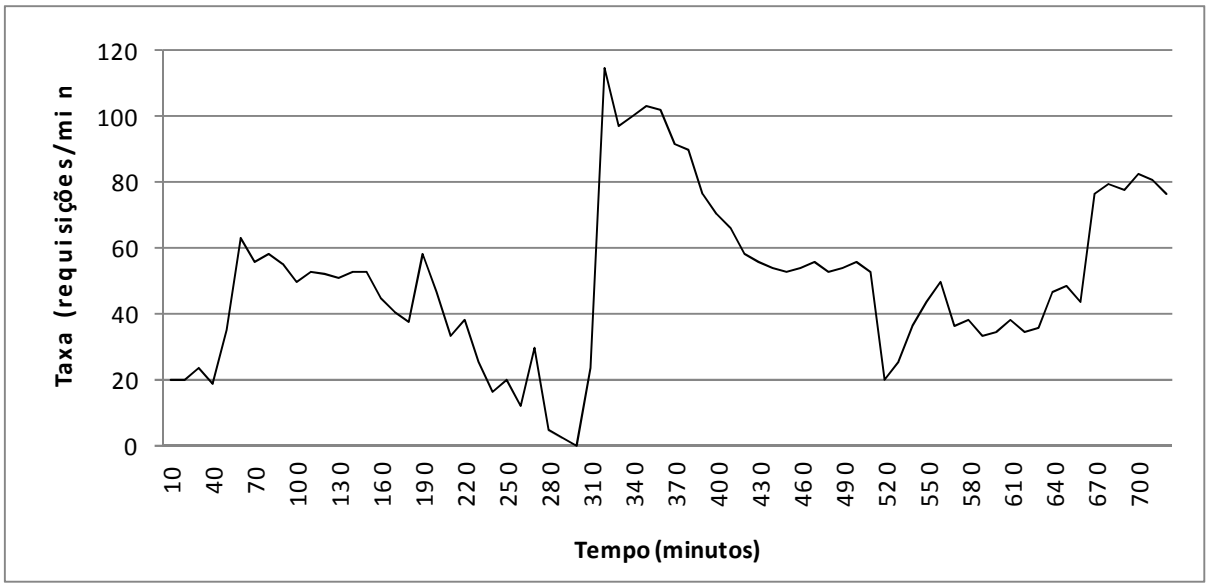


Figura 32 – Comportamento do *workload* com *logs* de acesso à página da NASA

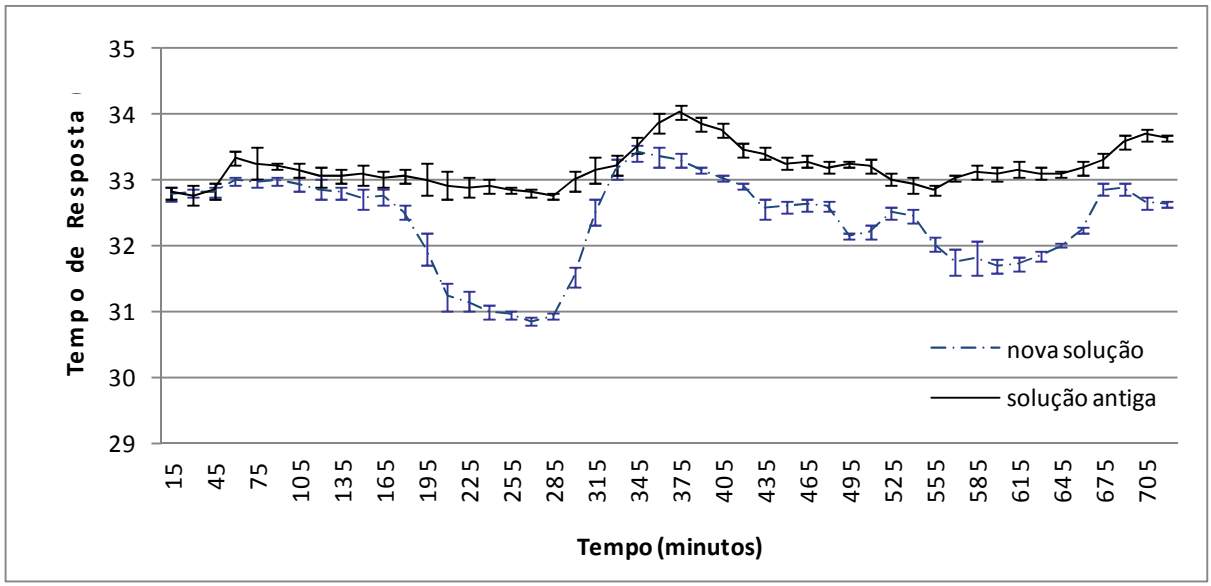


Figura 33 – Tempo de resposta para o *workload* da NASA

A Figura 33 apresenta o comportamento do tempo de resposta da aplicação para o cenário de carga utilizado neste teste, já a Figura 34 descreve a evolução do consumo de energia durante as 12 horas de execução do teste.

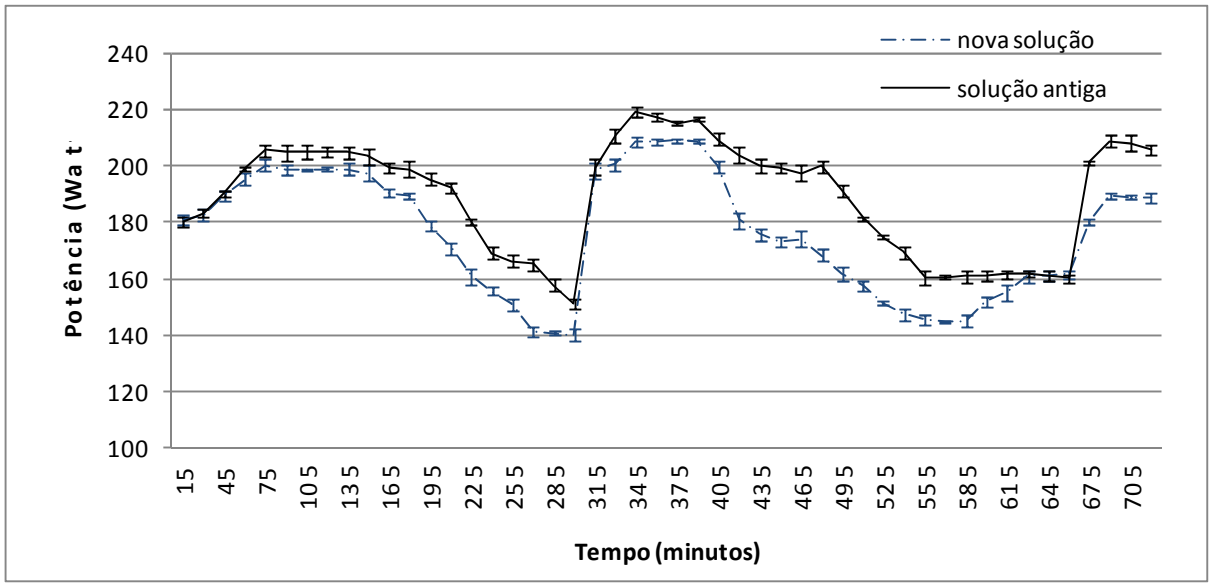


Figura 34 – Consumo de energia para o *workload* da NASA

Durante a execução do *workload* da NASA são verificadas sete transições entre as regiões de classificação do cluster. A primeira delas ocorre aos 50 minutos, quando o sistema migra da região 1 diretamente para a região 4. Com a adição repentina de recursos no ambiente, é possível observar uma elevação imediata do consumo de energia, enquanto que o tempo de resposta consegue manter-se estável, apesar do aumento da carga de requisições submetidas. Posteriormente, aos 135 minutos o sistema realiza uma transição da região 4 para a região 2, e em seguida aos 200 minutos é efetuada uma nova transição, desta vez da região 2 para a região 1. Neste momento, o sistema passa a operar com sua configuração mínima de recursos, e como conseqüência o valor de consumo de energia atinge seu piso aos 295 minutos registrando o valor de 140,3 Watts. Aos 305 minutos é observado o súbito aumento da quantidade de requisições submetidas ao *cluster*. Esta situação ocasiona uma transição da região 1 diretamente para a região 3, ou seja, a intervenção efetivada pela API reconfigura o sistema de sua configuração mínima para sua configuração máxima de recursos disponibilizados. Esta situação de oscilação elevada do *workload* é detectada e tratada pela nova política desenvolvida neste estudo e é responsável por evitar que o tempo de resposta

viole as restrições de operação, fato este que não ocorre na arquitetura original onde o tempo de resposta registra o valor de 33,71 ms aos 370 minutos de duração do teste.

O sistema permanece classificado na região 3 até os 410 minutos de teste, sendo possível verificar que durante o período que o sistema permanece na região 3 são observados os valores máximos de consumo de energia. Aos 410 minutos ocorre uma transição da região 3 para a região 5, iniciando o processo de queda dos valores de consumo de energia. Este processo é continuado durante o instante igual a 495 minutos, quando ocorre uma transição da atual região 5 diretamente para a região 2, levando o valor do consumo de energia para valores próximos ao mínimo observado durante todo o teste. Cabe ressaltar que o tempo de resposta também apresenta um comportamento decrescente desde os 410 minutos. Isto é observado em virtude da taxa de requisições representar um cenário de baixa/média carga. Por fim, aos 610 minutos há uma transição com origem na atual região 2 e destino na região 5, sem passar pela região intermediária 4. Esta última transição volta a elevar o valor do consumo de energia do ambiente, mas é capaz de evitar que o tempo de resposta ultrapasse o valor máximo estabelecido para o funcionamento da aplicação suportada pelo *cluster*.

Em relação à arquitetura original, a solução desenvolvida neste estudo efetuou uma redução média de 20,9% dos valores medidos de tempo de resposta e de 18,7% do consumo de energia do ambiente. A nova política de reconfiguração não permitiu violação das restrições de operação do tempo de resposta, e se mostrou capaz de se adaptar a qualquer tipo de cenário de carga, mesmo que o *workload* apresente variações significativas de carga em curtos intervalos de tempo, como foi o caso da variação observada nos instante próximo aos 305 minutos de teste.

4.5 Considerações sobre consumo de CPU

Embora o consumo de CPU não tenha sido utilizado como parâmetro para o processo de tomada de decisão da rede neural, apresentamos os gráficos referentes a esta métrica para facilitar a verificação das intervenções de gerenciamento de recursos efetivadas pela API. As Figuras 35 e 36 descrevem o consumo de CPU dos hospedeiros Itaipu (hospedeiro 1) e Ilha (hospedeiro 2) durante a execução dos *workloads* da Copa do Mundo de 98 e da NASA, respectivamente. Como cada hospedeiro possui dois núcleos de processadores que são ativados e inativados de acordo com as intervenções efetivadas junto ao *cluster*, os gráficos representam como consumo máximo de CPU o valor de 200%, que é referente aos dois

núcleos ativos. Desta forma, os valores acima de 100% indicam que há dois processadores ativos no momento.

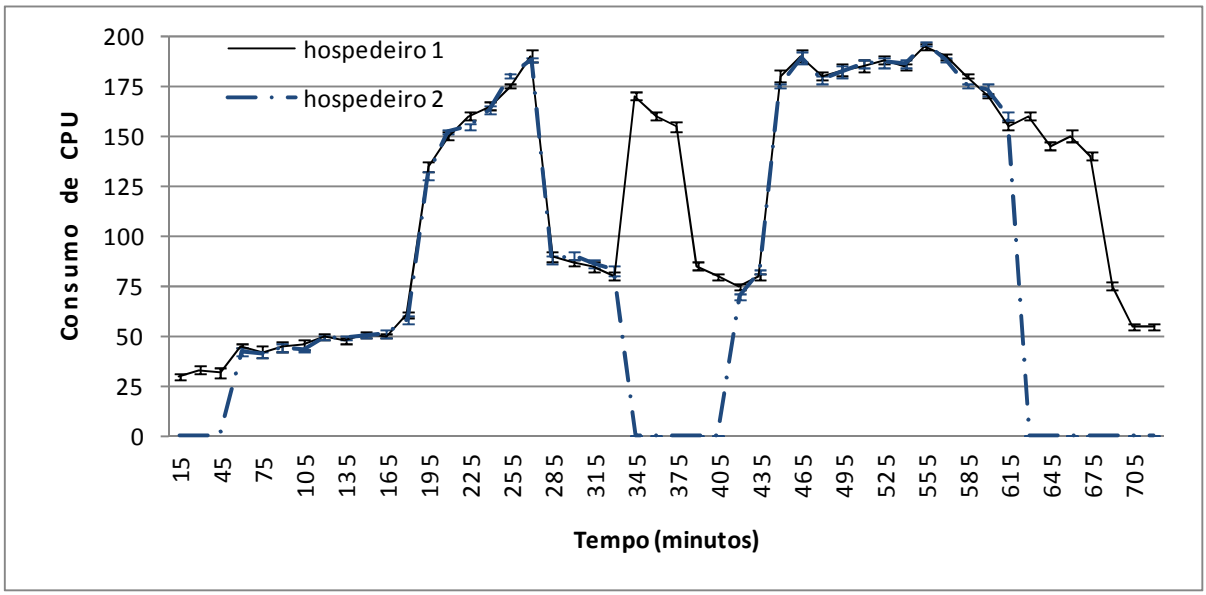


Figura 35 – Consumo de CPU dos servidores hospedeiros com o *workload* da Copa de 98

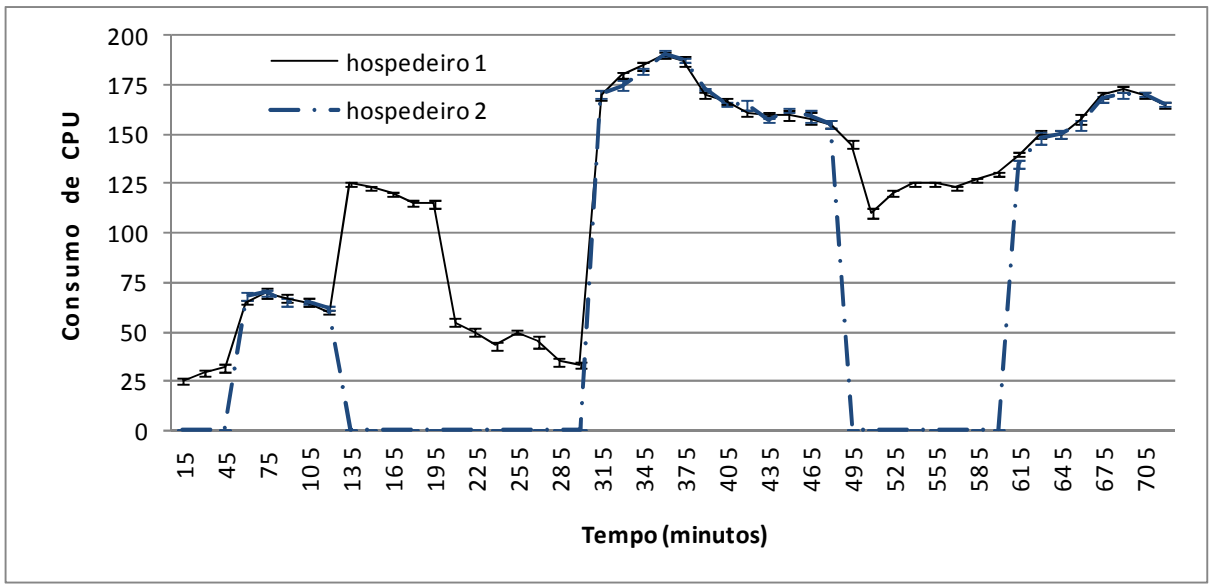


Figura 36 – Consumo de CPU dos servidores hospedeiros com o *workload* da NASA

É possível observar em ambos os gráficos que nos momentos onde os dois servidores hospedeiros estão ativos os valores de consumo de CPU de cada hospedeiro são semelhantes, pois a pequena diferença observada entre os dois valores presentes no gráficos está dentro do

intervalo de confiança utilizado. Esta semelhança demonstra que o elemento responsável pelo balanceamento de requisições do sistema, implementado no servidor Ipanema conforme indicado na Seção 3.3, consegue distribuir quase que igualmente entre os dois servidores hospedeiros das VMs as requisições submetidas ao *cluster*.

CAPÍTULO 5 CONCLUSÃO E TRABALHOS FUTUROS

Neste trabalho foi apresentada uma política de gerência e um sistema de controle de recursos de um *cluster* de servidores *web* hospedados em máquinas virtuais, que faz uso de uma rede neural artificial para subsidiar o processo de tomada de decisão das intervenções no ambiente de processamento. O objetivo do sistema desenvolvido é efetuar uma redução do consumo de energia do ambiente, mantendo o nível de qualidade de serviço dentro dos padrões estipulados para a aplicação. A arquitetura original utilizada como base para este trabalho foi adaptada para utilização da rede neural, e a política de reconfiguração foi refeita por completo de forma a utilizar as informações processadas pela rede neural.

Conforme discutido ao longo do texto, primeiramente foi realizada a validação da rede neural proposta, de forma a indicar a viabilidade da solução desenvolvida. Após a fase de validação, a rede neural foi configurada para receber em sua camada de entrada três indicadores de monitoração do ambiente: (i) tempo de resposta, (ii) consumo de energia e (iii) carga de requisições submetidas ao *cluster*. Assim, a rede neural é responsável por efetuar a análise e classificação dessas informações, produzindo como saída um conjunto de cinco regiões que estão associadas a estados distintos de operação do *cluster*, de acordo com as três métricas utilizadas. Realizada esta classificação, foram definidas as intervenções a serem efetivadas pela API junto ao *cluster* para reconfiguração do ambiente, disponibilizando a quantidade necessária de recursos conforme o estado de operação classificado pela rede neural. As ações de retirada e entrega de recursos ao ambiente visa evitar a ociosidade de dispositivos, levando assim a uma redução do consumo de energia, e cumprir as restrições impostas para a qualidade de serviço da aplicação.

Na avaliação desempenho, foi verificado que esta nova arquitetura é capaz de executar corretamente as reconfigurações, independente do *workload* apresentado. Desta forma, a solução proposta mostra-se apta a trabalhar com qualquer cenário de carga observado, pois a nova política de reconfiguração prevê transições entre qualquer estado de classificação do cluster, sem necessidade de transitar por estados intermediários. Além disso, observa-se que a presente solução apresentou um melhor desempenho perante a arquitetura original, efetuando uma redução mais expressiva do consumo de energia sem comprometer as restrições de operação para o tempo de resposta. Cabe ressaltar que o tempo de resposta observado com a utilização do sistema desenvolvido é, em média, menor do que aquele verificado pela

utilização da arquitetura original. Assim, é possível realizar intervenções que possam ajustar de forma eficiente o ambiente de processamento para o cenário de carga observado, sem submeter o cluster a passos intermediários de ajuste, tornando a reação do sistema mais ágil em relação à versão original.

Como trabalho futuro, sugere-se uma mudança na metodologia de treinamento da rede neural, de forma que a mesma seja treinada em tempo real para que seja possível identificar novos padrões de operação do cluster, caso os mesmos venham a surgir. Para esta nova abordagem, faz-se necessário uma análise detalhada do impacto deste novo tipo de treinamento na capacidade de resposta da rede neural, pois a fase de treinamento é realizada através de um processamento contínuo de dados. Caso não haja recursos computacionais disponíveis para esta tarefa, a rede neural pode não ser capaz de classificar os estados de operação quando solicitada, atrasando ou até inviabilizando a indicação de qual intervenção deve ser realizada no cluster em determinado momento. Para evitar este tipo de gargalo no processamento da rede neural, pode ser verificada a formação de um cluster de servidores físicos para processamento paralelo dos dados durante a etapa de treinamento. Esta solução se mostra interessante quando há um conjunto com grande quantidade de servidores físicos que hospedam as VM responsáveis pelo funcionamento da aplicação. Assim, o impacto causado pelo conjunto de servidores dedicados ao processamento da rede neural será relativamente pequeno perante o consumo global de energia de todo o ambiente.

Outro campo que pode ser explorado é o suporte a aplicações heterogêneas pelo conjunto de servidores virtuais, cada uma com características de comportamento distintas. Neste cenário, cada aplicação possui suas próprias restrições de operação para manutenção da qualidade de serviço, e o processo de gerência deve ser mostrar capaz de tratar a disponibilidade de recursos de acordo com a situação de cada aplicação. Para esta abordagem, torna-se imperativa a utilização de novas métricas para análise e processamento da rede neural, pois não é possível indicar apenas um valor de tempo de resposta, ou de carga de requisições, por exemplo.

Por fim, a possibilidade de configurar dinamicamente a quantidade de VMs ativas que suportam a aplicação também pode ser explorada. Como a infra-estrutura utilizada para desenvolvimento da presente solução possui a quantidade fixa de quatro servidores virtuais ativos, esta possibilidade não foi verificada no escopo deste trabalho. Entretanto, quando há disponibilidade de recursos físicos para manipulação de uma quantidade maior de VMs, a

criação de novas instâncias virtuais sob demanda se mostra uma alternativa interessante para manter o nível de qualidade de serviço em cenários de carga elevada ou, no caminho contrário, pode-se efetuar a exclusão de VMs quando o ambiente apresentar ociosidade de recursos em virtude de uma quantidade baixa de requisições submetidas ao *cluster*.

REFERÊNCIAS

[1] Intergovernmental Panel on Climate Change. Definitions and Methodological Options to Inventory Emissions from Direct Human-induced Degradation of Forests and Devegetation of Other Vegetation Types, IPCC National Greenhouse Gas Inventories Programme, Genebra, Suíça, Jan de 2006. v 4.

[2] Agência Nacional de Energia Elétrica. Matriz energética brasileira. Disponível em http://www.aneel.gov.br/aplicacoes/capacidadebrasil/OperacaoCapacidadeBrasil.asp. Acesso em: Jan de 2010.

[3] Maia S, Durão VS. Indústria ganha eficiência energética no pós-apagão. Jornal Valor Econômico; ed. 18/09/2007.

[4] Miranda MR. Eco-eficiência no datacenter. Disponível em: http://www.sun.com/eco. Acesso em: Dez de 2009.

[5] Levy MJ, Showhorn J. Datacenter Power Trends. In: 42th North American Network Operators Group Power Panel. San Jose, CA, USA, Fev de 2008.

[6] Niles S. Virtualization: Optimized power and cooling to maximize benefits. Disponível em: http://www.apc.com. Acesso em: Mai de 2009.

[7] Azevedo MV. Otimização de recursos e economia de energia em *cluster*s usando virtualização. 2010. 78 p. Dissertação de Mestrado – Faculdade de Engenharia, Universidade do Estado do Rio de Janeiro, Rio de Janeiro, Brasil, Jul de 2010.

[8] Megav I. Towards Automatic Management and Live Migration of Virtual Machines. 2007. 107 p. Master's thesis - Oslo University College (UiO / HiO), University of Oslo, Norway, Mai de 2007.

[9] Kusic D, Kephart J, Hanson J, Kandasamy N, Jiang G. Power and performance management of virtualized computing environments via lookahead control. In: IEEE International Conference on Autonomic Computing. Chicago, IL, USA, Jul de 2008.

[10] Ruth P, Rhee J, Xu D, Kennell R, Goasguen S. Autonomic Live Adaptation of Virtual Computational Environments in a Multi-Domain Infrastructure. In: IEEE International Conference on Autonomic Computing. Dublin, Ireland, Jun de 2006.

[11] Silva CS. Previsão de carga em um aglomerado de servidores *web* aplicada a economia de energia. 2010. 76 p. Dissertação de Mestrado – Instituto de Computação, Universidade Federal Fluminense, Niterói, Brasil, Mar de 2010.

[12] Petrucci V. A framework for supporting dynamic adaptation of Power-aware *web* Server *cluster*s. 2008. 61p. Dissertação de Mestrado – Instituto de Computação, Universidade Federal Fluminense, Niterói, Brasil, Jul de 2008.

[13] Rossi F. Alocação Dinâmica de Recursos no Xen. 2008. 70p. Dissertação de Mestrado Faculdade de Informática, Pontíficia Universidade Católica do Rio Grande do Sul, Porto Alegre, Brasil, Ago de 2008.

[14] Laaksonen J, Viitaniemi V, Koskela M. Application of self-organizing maps and automatic image segmentation to 101 object categories database. In: 4th International Workshop on Contet-Based Multimedia Indexing. Riga, Letônia, Jun de 2005.

[15] Yang Z, Laaksonen J. Interective retrieval in voice database using self-organizing maps.

In: 9th IAPR Conference on Machine Vision Applications. Tsukuba, Japão, Mai de 2005.

[16] Souza LS, Leite JB, Souza JC. Aplicação de redes neurais na construção de servidores *web* "verdes". Em: Conferencia Latinoamericana de Informática. Santa Fé, Argentina, Set de 2008.

[17] Simula O, Alhoniemi E, Hollmen J, Vesanto J. Monitoring and modeling of complex process using hierarchical self-organizing maps. 2008. 12p. Techinical Report – Laboratory of Computer and Information Science, Helsinki University of Techno*log*y.

[18] Lin MG. Metodologia para classificação de padrões de consumo de memória no Linux baseada em mapas auto-organizáveis. 2006. 76 p. Dissertação de Mestrado – Departamento de Ciência da Computação, Universidade Federal do Amazonas, Manaus, Brasil, Dez de 2006.

[19] Matthews JN, Dow EM, Deshane T, Hu W, Bongio J, Wilbur PF, Johnson B. Running Xen: A Hands-On Guide to the Art of Virtualization. Prantice Hall. 2008.

[20] Case P, Padegs A. Architecture of the IBM system/370. Jan de 1978; v. 21. n. 1. p. 73–96.

[21] Intel, Intel Virtualization Techno*log*y. Disponível em: http://www.intel.com/techno*log*y/virtualization. Acesso em: Mar de 2010.

[22] Advanced Micro Devices, AMD releases - pacifica specification for amd64 techno*log*y. Acesso em março de 2010. Disponível em: http://www.amd.com/us-en/*Web*lets. Acesso em: Jan de 2010.

[23] Smith JE, Nair R. The architecture of virtual machines. Computer, IEEE Computer Society Press; Mai de 2005. v. 38. n. 5. p. 32–38.

[24] Carnevale G. Microsoft Virtual PC. Disponível em: http://www.microsoft.com/brasil/technet/Colunas/GuilhermeCarnevale/VirtualPC.mspx . Acesso em Fev de 2010.

[25] OpenVZ, OpenVZ. Disponível em: http://wiki.openvz.org. Acesso em: Mar de 2010.

[26] Barham P, Dragovic B, Fraser K, Hand S, Harris T, Ho A et al. Xen and the Art of Virtualization. In: 19th ACM symposium on Operating systems principles, Nova York, USA, Out de 2003.

[27] Haykin S. Neural Networks. Prantice Hall. 2008.

[28] Honkela T. Self organizing maps in natural language processing. 1997. 98p. Doctor´s thesis. Helsinki University of Techno*log*y, Helsinki, Finland, Out de 1997.

[29] Costa JF, Netto MA. Segmentação de mapas auto-organizáveis com espaço de saída 3D. Revista Controle e Automação. Jun de 2007. v. 18. n. 2. p. 150-162.

[30] Danilo M. Estudo sobre o Mapa Auto-organizável de Kohonen em espaços 3D. Disponível em: http://ceia.labic.icmc.usp.br/mapeamento_ia/OC5.html. Acesso: em Dez de 2010.

[31] University of Stuttgart. Stuttgart neural network simulator. Disponível em: http://www.ra.cs.uni-tuebingen.de/SNNS. Acesso em: Jan de 2010.

[32] Hewlett Packard. HTTPERF. Disponível em: http://www.hpl.hp.com/research/linux/httperf. Acesso em: Mar de 2010.

[33] Midglei JJ. Autobench: a perl wrapper around httperf for automating benchmarking. Disponível em: http://www.xenoclast.org/autobench. Acesso em: Fev de 2010.

[34] CACTI. The complete rrdtool-based graphing solution. Disponível em: http://www.cacti.net. Acesso em: Abr de 2010.

[35] LBNL. The Internet traffic archive. Disponível em: http://ita.ee.lbl.gov/index.html. Acesso em: Jun de 2010.

APÊNDICE A – Funções da API

Função: hvmconnect
Funcionalidade: abre conexão com um hospedeiro
Parâmetros: objeto hvm do hospedeiro alvo
Retorno: (ponteiro) conexão para o hospedeiro ou nulo em caso de erro

Função: hvmgetvmcount
Funcionalidade: obtém quantidade de MVs em um hospedeiro
Parâmetros: objeto hvm do hospedeiro alvo
Retorno: (inteiro) quantidade de MVs

Função: hvmgetvmlist
Funcionalidade: obtém lista das MVs de um hospedeiro
Parâmetros: objeto hvm do hospedeiro alvo
Retorno: (string) lista das MVs ou nulo em caso de erro

Função: hvmgetspeed
Funcionalidade: obtém frequência atual de operação da CPU de um hospedeiro
Parâmetros: objeto hvm do hospedeiro alvo
Retorno: (inteiro) frequência atual em MHz ou -1 em caso de erro

Função: hvmmigratevm
Funcionalidade: faz migração ao vivo de uma MV entre dois hospedeiros
Parâmetros: nome da MV
objeto hvm do hospedeiro origem
objeto hvm do hospedeiro destino
Retorno: (inteiro) 0 em caso de sucesso ou -1 em caso de erro

Função: hvmsession
Funcionalidade: abre sessão com um hospedeiro (após uso de hvmconnect)
Parâmetros: objeto hvm do hospedeiro alvo
Retorno: (ponteiro) sessão com o hospedeiro ou nulo em caso de erro

Função: hvmsetspeed
Funcionalidade: define frequência de operação da CPU de um hospedeiro
Parâmetros: objeto hvm do hospedeiro alvo
frequência em MHz
Retorno: (inteiro) 0 em caso de sucesso ou -1 em caso de erro

Função: hvmsetcpuson
Funcionalidade: determina a quantidade de CPUs ou núcleos que devem estar ligados
Parâmetros: objeto hvm do hospedeiro alvo

quantidade de CPUs ou núcleos
Retorno: (inteiro) 0 em caso de sucesso ou -1 em caso de erro

Função: hvmsuspend
Funcionalidade: desativa um hospedeiro colocando-o em estado de espera
Parâmetros: objeto hvm do hospedeiro alvo
Retorno: (inteiro) 0 em caso de sucesso ou -1 em caso de erro

Função: hvmwakeup
Funcionalidade: seleciona um hospedeiro inativo e reativa o mesmo
Parâmetros: lista de objetos hvm dos hospedeiros
Retorno: (inteiro) número do hospedeiro disponível ou -1 em caso de erro

APÊNDICE B – Ações efetivadas pela API em cada região

Região 1

```
hvmssession(2);
hvmconnect (2);
hvmsession (1);
hvmconnect (1);
hvmsetcpuson (1,1);
hvmsetspeed (1, 1500);
retorno = hvmgetvmcount (2);
Se (retorno = 2)
    hvmgetvmlist (2);
    hvmmigratevm (vm1, 2, 1);
    hvmmigratevm (vm2, 2, 1);
FimSe
hvmsuspend (2);
```

Região 2 – carga crescente

```
hvmssession(2);
hvmconnect (2);
hvmsession (1);
hvmconnect (1);
hvmwakeup (2);
hvmgetsetccpuson (2,1);
hvmsetspeed (2, 1500);
hvmgetvmlist (1);
hvmmigratevm (vm3, 1, 2);
hvmmigratevm (vm4, 1, 2);
```

Região 2 – carga decrescente

```
hvmsession(2);
hvmconnect (2);
hvmsession (1);
hvmconnect (1);
hvmsetcpuson (1,2);
hvmsetspeed (1, 3000);
hvmgetvmlist (2);
```

```
hvmmigratevm (vm1, 2, 1);
hvmmigratevm (vm2, 2, 1);
hvmsuspend (2);
```

Região 3

```
hvmsession (2);
hvmconnect (2);
hvmsession (1);
hvmconnect (1);
retorno = hvmgetcvlist (2);
Se (retorno = nulo)
    hvmwakeup (2);
    hvmsetcpuson (2, 2);
    hvmsetspeed (2, 3000);
    hvmgetvmlist (1);
    hvmmigratevm (vm3, 1, 2);
    hvmmigratevm (vm4, 1, 2);
FimSe
hvmsetcpuson (1, 2);
hvmsetspeed (1, 3000);
```

Região 4 – carga crescente

```
hvmsession (2);
hvmconnect (2);
hvmsession (1);
hvmconnect (1);
retorno = hvmgetcvlist (2);
Se (retorno = nulo)
    hvmwakeup (2);
    hvmsetcpuson (2, 1);
    hvmsetspeed (2, 3000);
    hvmgetvmlist (1);
    hvmmigratevm (vm3, 1, 2);
    hvmmigratevm (vm4, 1, 2);
FimSe
hvmsetcpuson (1, 1);
hvmsetspeed (1, 3000);
```

Região 4 – carga decrescente

```
 hvmsession (2);
hvmconnect (2);
 hvmsession (1);
 hvmconnect (1);
 hvmsetcpuson (1, 1);
 hvmsetspeed (1, 1500);
hvmsetcpuson (2, 1);
hvmsetspeed (2, 1500);
```

Região 5

```
 hvmsession (2);
hvmconnect (2);
 hvmsession (1);
 hvmconnect (1);
 retorno = hvmgetcvlist (2);
 Se (retorno = nulo)
     hvmwakeup (2);
     hvmsetcpuson (2, 2);
     hvmsetspeed (2, 1500);
     hvmgetvmlist (1);
     hvmmigratevm (vm3, 1, 2);
     hvmmigratevm (vm4, 1, 2);
 FimSe
 hvmsetcpuson (1, 2);
 hvmsetspeed (1, 1500);
```