



UNIVERSIDADE DO VALE DO TAQUARI - UNIVATES  
CENTRO DE CIÊNCIAS EXATAS E TECNOLÓGICAS  
CURSO DE ENGENHARIA DA COMPUTAÇÃO

**ANÁLISE DE CONSUMO ENERGÉTICO EM UM CLUSTER DE ALTA  
DISPONIBILIDADE UTILIZANDO RED HAT ENTERPRISE LINUX**

Marco Antônio Arnhold

Lajeado, julho de 2019

Marco Antônio Arnhold

## **ANÁLISE DE CONSUMO ENERGÉTICO EM UM CLUSTER DE ALTA DISPONIBILIDADE UTILIZANDO RED HAT ENTERPRISE LINUX**

Monografia desenvolvida na disciplina de Trabalho de Conclusão de Curso II, do curso de Engenharia da Computação, da Universidade do Vale do Taquari – Univates, como requisito para obtenção do título de Bacharel em Engenharia da Computação.

Orientador: Prof. Me. Luis Antônio Schneiders

Lajeado, julho de 2019

## **AGRADECIMENTOS**

Agradeço, primeiramente, à minha mãe, Cléria, por todo o apoio e incentivo, não somente durante este trabalho, mas durante todos os anos de curso.

À minha amiga Ana Emília, pela amizade, pelo apoio emocional e pelas dicas na revisão e na formatação deste trabalho.

Aos meus amigos Vinícius e Lucas, também por sua amizade, apoio emocional e contribuições para o desenvolvimento deste trabalho.

Ao meu orientador, Prof. Me. Luis Antonio Schneiders, por todo o apoio e conhecimentos compartilhados durante o trabalho, além de me propiciar a trabalhar com um ambiente novo e fora do meu cotidiano.

Ao Marlon, do Laboratório de Automação da Univates, pelo suporte prestado na infraestrutura do datacenter.

Aos colegas de curso, pela troca de informações, dicas e apoio em aula durante a execução deste trabalho.

## RESUMO

Os clusters computacionais, os quais podem apresentar baixo custo e grande escalabilidade, vêm se tornando ótimas alternativas para cenários empresariais que exigem alta disponibilidade. Para atender a tais cenários, muitas vezes, a eficiência energética destes arranjos é desconsiderada, levando estes a consumirem quantidades elevadas de energia, o que gera grande impacto nos custos operacionais do cluster, bem como no meio ambiente. O presente trabalho tem por objetivo analisar o grau de eficiência energética de um cluster computacional de alta disponibilidade, voltado à provisão de máquinas virtuais como recursos. A fim de possibilitar a análise, foi configurado um cluster de dois nós, composto por servidores do tipo blade, nos quais o sistema operacional Red Hat Enterprise Linux (RHEL) 7.6 e a ferramenta de virtualização *Kernel Virtual Machine* (KVM) foram instalados. Foram efetuados experimentos sob diferentes cargas de trabalho, com o uso de softwares de *benchmark* e *stress*, sendo o consumo de energia elétrica mensurado usando ferramentas de gerenciamento do ambiente, como o *Integrated Dell Remote Access Controller* (iDRAC). Por meio do uso dessas ferramentas, de técnicas de virtualização e do pacote de softwares providos pelo RHEL, foi possível analisar a eficiência energética do arranjo, bem como evidenciar sua capacidade de prover um ambiente de alta disponibilidade de forma eficaz.

**Palavras-chave:** Cluster. Eficiência energética. Alta disponibilidade.

## ABSTRACT

Computational clusters, which can present low cost and high scalability, are becoming great alternatives for scenarios that require high availability. In order to meet such scenarios, the energy efficiency of these arrangements is often disregarded, causing them to consume high amounts of energy, generating a great impact on the operational costs of the cluster, as well as on the environment. The present paper aims to analyze the degree of power efficiency of a high availability computational cluster, focused on the provision of virtual machines as resources. In order to enable the analysis, a two-node cluster consisting of blade-type servers was configured, under which the Red Hat Enterprise Linux (RHEL) 7.6 operational system and the Kernel Virtual Machine (KVM) virtualization framework were installed. Experiments were carried out under different workloads, using benchmark and stress software, being the electrical energy consumption measured using infrastructure management tools, such as the Integrated Dell Remote Access Controller (iDRAC). Through the use of these tools, virtualization techniques and the software package provided by RHEL, it was possible to analyze the power efficiency of the arrangement, as well as to demonstrate its ability to provide a high availability environment in an effective way.

**Key words:** Cluster. Power efficiency. High Availability.

## LISTA DE ILUSTRAÇÕES

### LISTA DE FIGURAS

Figura 1 – Exemplos de disposições genéricas de multiprocessadores .....	22
Figura 2 – Disposições genéricas de multicomputadores .....	23
Figura 3 – Um sistema distribuído organizado como <i>middleware</i> .....	26
Figura 4 – Funcionamento simplificado de um <i>hub</i> .....	28
Figura 5 – Funcionamento simplificado de um <i>switch</i> .....	29
Figura 6 – Cluster de quatro nós compartilhando o hardware de rede e armazenamento .....	31
Figura 7 – Árvore de dependências de um servidor SQL .....	34
Figura 8 – Cluster simétrico .....	35
Figura 9 – Cluster assimétrico .....	36
Figura 10 – Representação de um cluster expandido com nós especializados para monitoramento, servidor NFS e servidor de E/S .....	37
Figura 11 – Cluster de alta disponibilidade com dois nós .....	39
Figura 12 – Arquitetura protocolar de uma grid .....	46
Figura 13 – Arquitetura de grid proposta por Tony .....	47
Figura 14 – Representação das topologias de intragrid, extragrid e intergrid .....	49
Figura 15 – Interface gráfica do RHEL com a janela "Sobre" aberta .....	50
Figura 16 – Princípio básico de máquinas virtuais .....	52
Figura 17 - Representação dos dois modos de implementação de máquinas virtuais .....	54
Figura 18 – Arquitetura da KVM .....	55
Figura 19 – Cluster desenvolvido com computadores de placa única .....	67
Figura 20 – Rack do data center educacional da Univates .....	71
Figura 21 – Chassi PowerEdge M1000e, destacando os dois nós do cluster configurado .....	72
Figura 22 – Tela Inicial do iDRAC 7 .....	73
Figura 23 – Interface Gráfica do Editor de Conexões do Network Manager .....	75
Figura 24 – Filtro “sshd” do Fail2Ban, apresentando os IPs bloqueados no momento .....	77
Figura 25 – SAN Dell EqualLogic PS6010 utilizada .....	78
Figura 26 – Interface web do pcs acessada através do IP virtual .....	84

Figura 27 – Interface gráfica do <i>Virtual Machine Manager</i> com a tela de criação de uma nova VM .....	85
Figura 28 – Utilitário <i>crm_mon</i> em execução .....	88
Figura 29 – Esquema da configuração final do cluster "kvm_ha" .....	89
Figura 30 – Software HeavyLoad em execução na VM 5 .....	96
Figura 31 – Ferramenta <i>top</i> em execução no momento do teste .....	99
Figura 32 – AB em execução na VM 5, juntamente do Gerenciador de Tarefas.....	102
Figura 33 – Relatórios finais do AB executado nas VMs 1 e 5.....	103
Figura 34 – Saída da ferramenta <i>top</i> no momento de execução dos testes com AB .....	104
Figura 35 – Listagem de fontes do chassi apresentada no CMC.....	108
Figura 36 – Listagem de fontes do chassi exibidas no CMC.....	109

## LISTA DE GRÁFICOS

Gráfico 1 – Custos anuais em um data center com o passar dos anos .....	61
Gráfico 2 – Comparação da Performance por Watt entre cada <i>hypervisor</i> e o <i>baseline</i> .....	69
Gráfico 3 – Linha do tempo de inicialização do ambiente e migração de recurso <i>VirtualDomain</i> .....	91
Gráfico 4 – Migração em situação de emergência - <i>timeout</i> 120 s.....	92
Gráfico 5 – Migração em situação de emergência - <i>timeout</i> 300 s.....	93
Gráfico 6 – Consumo elétrico do cluster com HeavyLoad.....	97
Gráfico 7 – Temperaturas do cluster com HeavyLoad .....	98
Gráfico 8 – Consumo elétrico do cluster com Apache Benchmark.....	105
Gráfico 9 – Temperaturas do cluster com Apache Benchmark .....	106

## LISTA DE TABELAS

Tabela 1 – Tecnologias de interconexão nos 30 computadores “mais verdes” da Green500 com o passar dos anos .....	30
Tabela 2 – Os dez supercomputadores mais eficientes do mundo, segundo a lista Green500 de junho de 2018.....	62
Tabela 3 – Distribuição dos recursos do hospedeiro entre as VMs.....	86

## LISTA DE ABREVIATURAS E SIGLAS

10-GbE	10 Gigabit Ethernet
AB	Apache Benchmark
ABI	Application Binary Interface
API	Application Programming Interface
CC	Cache Coherent
CLUMPS	Cluster of SMPs
CLVM	Clustered Logical Volume Manager
CLVMD	Cluster Logical Volume Manager Daemon
CMC	Chassis Management Controller
COTS	Commodity Off the Shelf
COW	Cluster of Workstations
CPU	Central Processing Unit
CRM	Cluster Resource Manager
DC	Designated Controller
DCiE	Data Center infrastructure Efficiency
DLM	Distributed Lock Manager
DNS	Domain Name System
EPA	Environmental Protection Agency
EPEAT	Electronic Product Environmental Assessment Tool
EUA	Estados Unidos da América
FLOP	Floating-point Operations Per Second

GB	Gigabyte
Gbps	Gigabits/segundo
Gflop/s	Gigaflop/segundo
GFS2	Global File System 2
GHz	Giga-hertz
GNOME	GNU Network Object Model Environment
GRAM	Grid Resource Access and Management
GUI	Graphic User Interface
GW	Gigawatt
HA	High Availability
HPC	High Performance Computing
HPL	High Performance Linpack
iDRAC	Integrated Dell Remote Access Controller
IP	Internet Protocol
IPMI	Intelligent Platform Management Interface
IPS	Intrusion Prevention System
iSCSI	Internet Small Computer System Interface
JVM	Java Virtual Machine
KVM	Kernel Virtual Machine
kWh	Quilowatt-hora
LAN	Local Area Network
LV	Logical Volume
LVM	Logical Volume Manager
Mbps	Megabits/segundo
MIMD	Multiple Instruction Multiple Data
MISD	Multiple Instruction Single Data
NASA	North American Space Agency
NCC	Non-Cache Coherent
NIC	Network Interface Controller
NOW	Network of Workstations
NUMA	Non-Uniform Memory Access
OSCAR	Open Source Cluster Application Resources

OV	Organização Virtual
PDU	Power Distribution Unit
PDU	Power Distribution Unit
PUE	Power Usage Efficiency
PV	Physical Volume
QEMU	Quick Emulator
QoS	Quality of Service
RAID	Redundant Array of Independent Disks
RDP	Remote Desktop Protocol
RHEL	Red Hat Enterprise Linux
RHV	Red Hat Virtualization
RPM	Rotações por Minuto
SAN	Storage Area Network
SAS	Serial Attached SCSI
SATA	Serial Advanced Technology Attachment
SCSI	Small Computer System Interface
SFP+	Enhanced Small Form-factor Pluggable
SIMD	Single Instruction Multiple Data
SISD	Single Instruction Single Data
SMP	Symmetric Multi-Processing
SQL	Structured Query Language
SSH	Secure Shell
SSI	Single System Image
STONITH	Shoot The Other Node In The Head
TB	Terabyte
TCO	Total Cost of Ownership
TI	Tecnologia da Informação
UMA	Uniform Memory Access
VG	Volume Group
VLAN	Virtual LAN
VM	Virtual Machine
VMM	Virtual Machine Manager

WAN	Wide Area Network
Ws	Watt-segundo
XML	Extensible Markup Language

## SUMÁRIO

<b>1 INTRODUÇÃO</b> .....	<b>14</b>
1.1 Tema .....	15
1.1.1 Delimitação do tema .....	15
1.2 Problema .....	16
1.3 Objetivos .....	16
1.3.1 Objetivo geral .....	16
1.3.2 Objetivos específicos .....	16
1.4 Justificativa .....	17
1.5 Estrutura do trabalho .....	18
<b>2 REFERENCIAL TEÓRICO</b> .....	<b>19</b>
2.1 Arquiteturas computacionais .....	19
2.1.1 Multiprocessadores .....	20
2.1.2 Multicomputadores .....	22
2.2 Sistemas distribuídos .....	23
2.2.1 Definição dos sistemas distribuídos .....	25
2.2.2 Rede e interconexão .....	27
2.2.2.1 Dispositivos convencionais de interconexão .....	27
2.2.2.2 Dispositivos de interconexão de alto desempenho .....	29
2.3 Clusters .....	30
2.3.1 Objetivos de um cluster .....	32
2.3.2 Abstrações de um cluster .....	32
2.3.2.1 Nó .....	32
2.3.2.2 Recurso .....	33
2.3.2.3 Dependência de recursos .....	33
2.3.2.4 Grupo de recursos .....	34
2.3.3 Estrutura de um cluster .....	34
2.3.3.1 Cluster simétrico .....	35
2.3.3.2 Cluster assimétrico .....	36
2.3.4 Aplicações-alvo .....	37
2.3.4.1 Cluster de alto desempenho .....	38
2.3.4.2 Cluster de alta disponibilidade .....	38
2.3.4.3 Cluster de balanceamento de carga .....	39

2.3.5 Tipos de computadores empregados.....	40
2.3.5.1 NOW / COW.....	40
2.3.5.2 Beowulf.....	40
2.3.5.3 CLUMPS.....	41
2.3.6 Tipos de nós.....	42
2.3.6.1 Cluster homogêneo.....	42
2.3.6.2 Cluster heterogêneo.....	42
2.3.7 Categorias dos clusters.....	43
2.3.8 Software e gerenciamento.....	44
2.4 Grids.....	44
2.4.1 Arquitetura.....	45
2.4.2 Topologia.....	48
2.5 Red Hat Enterprise Linux.....	49
2.6 Virtualização.....	50
2.6.1 Modos de implementação.....	52
2.6.2 KVM.....	54
2.6.2.1 Migração em tempo real.....	55
2.6.2.2 KVM no Red Hat Enterprise Linux.....	56
2.7 Eficiência energética.....	56
2.7.1 Green IT.....	57
2.7.1.1 Energy Star.....	58
2.7.1.2 EPEAT.....	58
2.7.1.3 80 Plus.....	59
2.7.1.4 Green Grid.....	59
2.7.2 Green500.....	60
3 PROCEDIMENTOS METODOLÓGICOS.....	63
3.1 Quanto à natureza.....	63
3.2 Quanto à abordagem.....	63
3.3 Quanto aos objetivos.....	64
3.4 Quanto aos procedimentos técnicos.....	65
3.5 Coleta de dados.....	66
3.6 Análise dos dados.....	66
3.7 Trabalhos relacionados.....	67
4 CONFIGURAÇÃO DO AMBIENTE.....	70
4.1 Instalação do Red Hat Enterprise Linux 7.6.....	72
4.1.1 Configuração de rede e segurança.....	74
4.1.2 Mapeamento do armazenamento SAN.....	77
4.2 Criação e configuração do cluster.....	79
4.2.1 Recursos do cluster.....	80
4.2.1.1 Isolamento do cluster – <i>fencing</i> .....	80
4.2.1.2 Sistema de arquivos compartilhado - GFS2.....	81
4.2.1.3 Endereço de IP virtual.....	84
4.2.1.4 Máquinas virtuais – <i>VirtualDomain</i> .....	85
4.2.2 Panorama do cluster configurado.....	88
5 EXPERIMENTOS E ANÁLISE DOS RESULTADOS.....	90

<b>5.1 Experimento 1 – Tempos para inicialização do cluster e migração de recursos</b>	<b>90</b>
5.1.1 Resultados	91
5.1.2 Análise dos resultados	93
<b>5.2 Experimento 2 – Consumo de energia elétrica durante teste de stress com HeavyLoad</b>	<b>95</b>
5.2.1 Resultados	97
5.2.2 Análise dos resultados	98
<b>5.3 Experimento 3 – Consumo energético durante execução do Apache Benchmark</b>	<b>100</b>
5.3.1 Resultados	101
5.3.2 Análise dos resultados	106
<b>5.4 Consumo de energia elétrica do chassi de blades</b>	<b>107</b>
<b>6 CONSIDERAÇÕES FINAIS</b>	<b>110</b>
<b>REFERÊNCIAS</b>	<b>112</b>

## 1 INTRODUÇÃO

Devido ao contínuo desenvolvimento da indústria da computação e de recursos tecnológicos, os quais permitem a criação de diversos tipos de aplicações, é cada vez mais elevado o número de organizações públicas ou privadas que dependem de sistemas computacionais para seu correto funcionamento, ou até para prover seus serviços. Desde lojas virtuais até sistemas bancários e caixas de lojas, os computadores integram papéis críticos no dia a dia da sociedade, sendo que eventuais indisponibilidades podem causar sérios problemas e prejuízos (COSTA, 2009).

Com a crescente necessidade de manter estes serviços críticos em funcionamento e sem interrupções, o emprego do conceito de alta disponibilidade se torna essencial. Esse se caracteriza pela capacidade de um sistema se manter disponível, independentemente de eventuais falhas. Para possibilitar tal característica, a redundância de infraestrutura e eliminação de pontos únicos de falha são elementos chave da configuração do ambiente (PEREIRA JÚNIOR, 2010).

O emprego de agregados computacionais, ou clusters, se mostra cada vez mais comum para prover um ambiente de alta disponibilidade, seja em grandes datacenters ou até em pequenas empresas e instituições de pesquisa. Neste tipo de arranjo, ao menos dois computadores formam o cluster, sendo um deles o nó principal ativo e o outro, o nó de *failover*, o qual fica ocioso. Caso venha a ocorrer uma eventual falha no nó principal, o nó ocioso irá assumir, garantindo a disponibilidade dos serviços (REIS et al., [2011?]).

No entanto, a operação de ambientes de alta disponibilidade, por vezes, pode vir a consumir quantidades elevadas de energia elétrica, caso esses não sejam planejados e otimizados, de uma forma energeticamente eficiente. Estima-se que 63% do custo total de propriedade, ou *Total Cost of Ownership* (TCO), de um data center é resultado de gastos com equipamentos de energia e resfriamento, bem como com custos de energia. Levando em consideração somente os data centers, estes foram responsáveis pelo consumo de 61 bilhões de quilowatts-hora (kWh) no ano de 2006, somente nos EUA, o que seria suficiente para abastecer em torno de 5,8 milhões de residências (FELLER; MORIN; LEPRINCE, 2010).

Dessa forma, os custos cada vez mais elevados da energia elétrica aliados ao impacto ambiental causado pela sua geração, tornaram a eficiência energética um dos principais pontos de atenção no desenvolvimento de novos sistemas computacionais (FELLER; MORIN; LEPRINCE, 2010).

Presume-se que 35% a 50% da energia consumida pelos nós de um cluster computacional decorre das suas CPUs (*Central Processing Unit*), seguido pelos módulos de memória. O uso de componentes de baixo consumo aliado a técnicas para gerenciamento dinâmico de energia, como o balanceamento de carga e virtualização, permite uma melhora significativa na eficiência energética do arranjo, possibilitando diminuição de gastos também com refrigeração (VALENTINI et al., 2011).

## **1.1 Tema**

O tema do presente trabalho está relacionado com a melhoria da eficiência energética de clusters computacionais de alta disponibilidade e de alto desempenho.

### **1.1.1 Delimitação do tema**

No presente trabalho, foi configurado um ambiente de cluster computacional de alta disponibilidade com foco em virtualização, em que os nós executam uma distribuição Linux, chamada de Red Hat Enterprise Linux (RHEL) versão 7.6. O foco

do estudo é a análise do consumo energético do arranjo computacional montado e, com base nesses resultados e na pesquisa bibliográfica, a aplicação de possíveis otimizações que venham a aprimorar a sua eficiência energética.

## **1.2 Problema**

O consumo de energia é uma preocupação importante no contexto de clusters, visto que influencia, de forma direta, as necessidades de refrigeração e estrutura deste, gerando impacto expressivo nos custos de manutenção, bem como no meio ambiente como um todo (PINHEIRO et al., 2001). Com base nessa afirmativa, pretende-se responder à seguinte pergunta: É possível aumentar a eficiência energética de um cluster computacional sem comprometer seu desempenho e sua disponibilidade?

## **1.3 Objetivos**

Esta subseção descreve o objetivo geral e os objetivos específicos do presente trabalho.

### **1.3.1 Objetivo geral**

O objetivo deste trabalho é analisar o grau de eficiência energética de um cluster computacional de alta disponibilidade, voltado à provisão de máquinas virtuais como recursos.

### **1.3.2 Objetivos específicos**

Os objetivos específicos deste trabalho são:

- Configurar um cluster computacional de alta disponibilidade de dois nós;

- Utilizar a ferramenta de virtualização KVM (*Kernel Virtual Machine*), presente na distribuição do Linux Red Hat Enterprise Linux versão 7.6;
- Mensurar o consumo energético do cluster sob diferentes cargas de trabalho;
- Analisar a eficiência energética do cluster;
- Identificar e aplicar possíveis otimizações na configuração do cluster, a fim de otimizar sua eficiência energética.

#### **1.4 Justificativa**

Este trabalho justifica-se pela constante e crescente necessidade de buscar métodos e meios que aumentem a eficiência energética de arranjos computacionais como clusters, a fim de minimizar o impacto financeiro e ambiental causado por estes.

Para o acadêmico, acrescenta em conhecimentos sobre a arquitetura dos clusters, principalmente em aplicações dedicadas à alta disponibilidade e virtualização. Também agrega conhecimentos a respeito do gerenciamento energético de estruturas deste tipo, bem como o impacto causado por elas. Outro ponto de destaque é o emprego do sistema operacional RHEL 7.6, uma das mais conceituadas distribuições do Linux para o mercado corporativo e clusters de larga escala.

O projeto também pode ser relevante para a instituição acadêmica na qual o acadêmico está inserido, devido ao emprego pioneiro do sistema operacional RHEL 7.6 como sistema do cluster a ser configurado e analisado, expondo funcionalidades e vantagens deste. Aos demais estudantes e comunidade em geral, espera-se evidenciar o funcionamento e eficiência energética de uma configuração deste tipo, demonstrando possíveis otimizações que poderão ser aplicadas em outros projetos e visando à criação de configurações mais sustentáveis e com um aproveitamento de recursos inteligente.

## 1.5 Estrutura do trabalho

O presente trabalho está estruturado em seis capítulos.

O capítulo 1 consiste em uma introdução ao tema da computação de alto desempenho e alta disponibilidade e à sua questão energética. O capítulo também integra o tema, o problema, os objetivos e a justificativa.

O capítulo 2 refere-se ao referencial teórico utilizado como base para o desenvolvimento dos objetivos propostos, compreendendo conteúdos como arquitetura de computadores, sistemas distribuídos, clusters, grids, virtualização e eficiência energética em sistemas computacionais.

No capítulo 3, está descrita a metodologia de pesquisa empregada para o desenvolvimento do projeto.

O capítulo 4 apresenta as etapas para a configuração do ambiente de cluster desenvolvido, detalhando os comandos e a estrutura utilizados.

O capítulo 5 consiste na descrição e resultados dos experimentos realizados, juntamente com a análise destes.

O capítulo 6 evidencia as conclusões obtidas após a realização deste trabalho.

Por fim, são apresentadas as referências bibliográficas utilizadas para a fundamentação teórica deste trabalho.

## 2 REFERENCIAL TEÓRICO

Neste capítulo, são abordadas, inicialmente, as arquiteturas computacionais utilizadas em sistemas distribuídos, partindo para uma definição deste tipo de sistema e suas principais características. Além disso, são apresentados os arranjos computacionais de clusters e grids, com maior foco no primeiro, explanando seus objetivos, arquitetura, categorias e aplicações alvo.

Também é explanado o conceito de virtualização e uma das ferramentas específicas que será utilizada no projeto, o *Kernel Virtual Machine* (KVM). Por fim, são detalhadas algumas das principais iniciativas de entidades de Tecnologia da Informação (TI) para a otimização da eficiência energética.

### 2.1 Arquiteturas computacionais

Dada a enorme quantidade de arquiteturas de computadores existentes, diversas taxonomias surgiram de forma a tentar padronizar adequadamente as especificações desses diversos sistemas (DANTAS, 2005).

Flynn (1972) propôs a classificação da estrutura de sistemas de computadores levando em conta possíveis modelos de interação entre instruções e informações. As quatro classificações estabelecidas por ele são:

- *Single Instruction Single Data (SISD)*: computadores que executam uma instrução sob um fluxo de dados por vez. Os computadores pessoais com processadores convencionais são um exemplo.
- *Single Instruction Multiple Data (SIMD)*: computadores que executam uma única instrução por vez, mas sob um conjunto de dados múltiplos, comumente armazenados em vetores ou *arrays*. Máquinas como ILLIAC IV, da Universidade de Illinois, trabalhavam dessa maneira.
- *Multiple Instruction Single Data (MISD)*: são máquinas que executam múltiplas instruções sobre um conjunto único de dados e suas derivações. De acordo com Dantas (2005), este tipo de arquitetura nunca foi efetivamente utilizado em máquinas modernas.
- *Multiple Instruction Multiple Data (MIMD)*: computadores com esta arquitetura possuem vários processadores, podendo executar múltiplas instruções sob múltiplos fluxos de informações. Alguns exemplos são os computadores da linha SP da IBM (DANTAS, 2005) e os computadores Exemplar e SuperDome da HP (PŁAŻEK; BANAS; KITOWSKI, 2001).

Os computadores do tipo MIMD possuem diferenças entre si, de acordo com a forma como seus processadores e memórias são interligados. Comumente, são classificados como multiprocessadores e multicomputadores (DANTAS, 2005).

### **2.1.1 Multiprocessadores**

Arquiteturas do tipo multiprocessadores se caracterizam por diversos processadores que compartilham uma mesma memória ou um conjunto delas. Uma vez que a memória e os processadores estão fortemente ligados entre si através de uma estrutura de interconexão, afirma-se que este tipo de arquitetura é fortemente acoplada (DANTAS, 2005).

Segundo Hill, Jouppi e Sohi (2000), muitos multiprocessadores permitem que seus processadores possuam caches privados, que armazenam cópias de dados ainda presentes nos endereços originais de memória. Devido à estrutura

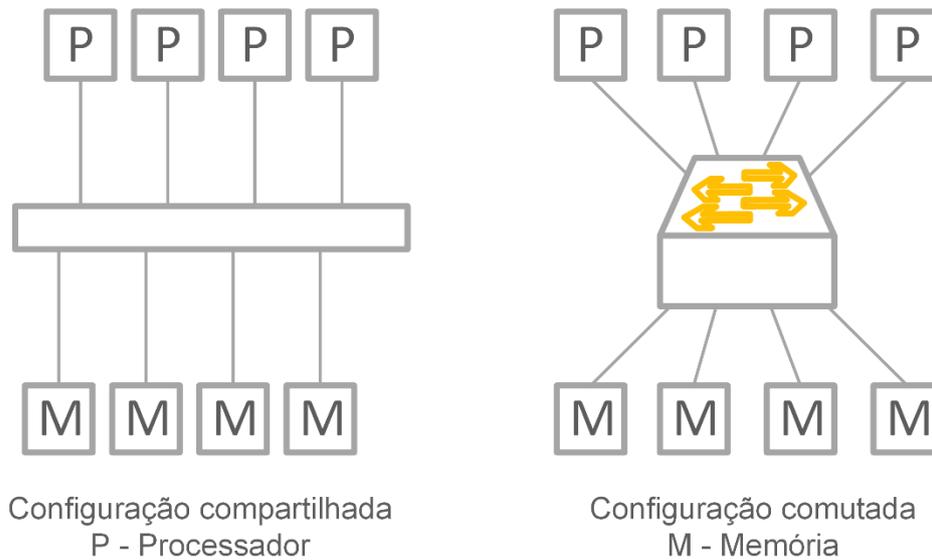
compartilhada e à quantidade de processadores, garantir que todos recebam e atualizem os dados mais recentes gera problemas de coerência de cache, os quais, comumente, podem ser resolvidos com a adoção de protocolos específicos. Denominam-se de *Cache Coherent* (CC) os sistemas que adotam ao menos um protocolo de coerência de cache e de *Non-Cache Coherent* (NCC) os que não o fazem.

Ainda segundo Hill, Joupe e Sohi (2000), outra diferenciação que deve ser notada nos multiprocessadores é a forma pela a qual a memória é acessada, que pode ser:

- Acesso Uniforme à Memória (UMA - *Uniform Memory Access*): neste tipo, todos os acessos à memória ocorrem com um atraso similar;
- Acesso não-Uniforme à Memória (NUMA - *Non-Uniform Memory Access*): neste tipo, os acessos à memória ocorrem com atrasos diferentes. Um exemplo é o acesso à memória local, o qual ocorre rapidamente, diferente de quando for necessário acessar a memória de qualquer outro, que pode levar mais tempo (TANEMBAUM; STEEN, 2002).

Quanto à sua interconexão, os multiprocessadores podem ser interconectados através de um barramento, o qual possibilita fácil configuração, ou por meio de um equipamento comutador (*switch*), que permite uma maior escalabilidade de níveis (DANTAS, 2005). Na Figura 1, podemos observar dois exemplos de multiprocessador, cada um representado um tipo de interconexão.

Figura 1 – Exemplos de disposições genéricas de multiprocessadores

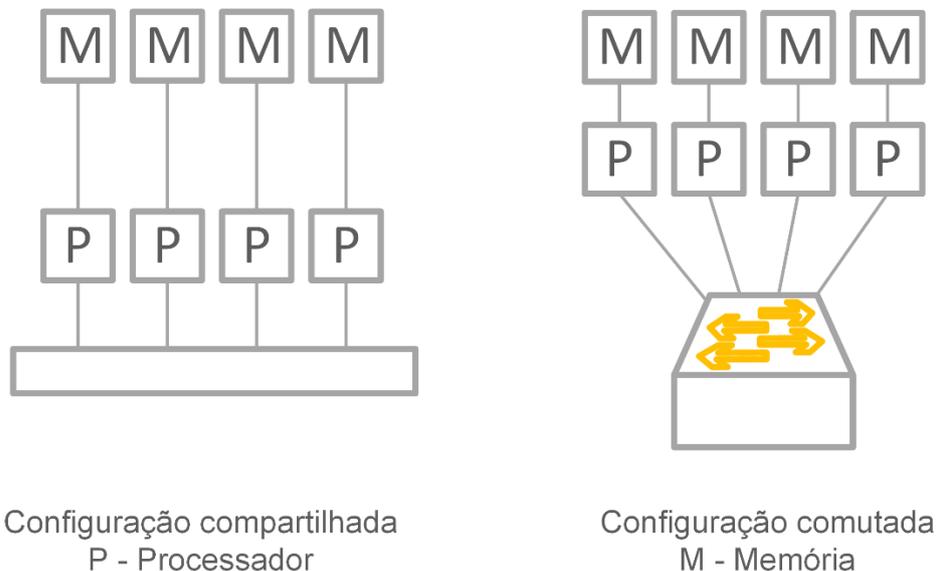


Fonte: Adaptada pelo autor, com base em Dantas (2005).

### 2.1.2 Multicomputadores

Arquiteturas do tipo multicomputadores se caracterizam por serem fracamente acopladas, em que cada processador possui sua memória local. Não existindo uma partilha forte, a comunicação ocorre somente pela troca de mensagens entre processos em execução nos processadores (DANTAS, 2005). A Figura 2 representa uma estrutura genérica multicomputador.

Figura 2 – Disposições genéricas de multicomputadores



Fonte: Adaptada pelo autor, com base em Dantas (2005).

De acordo com Hill, Jouppe e Sohi (2000), cada nó pode possuir um ou mais processadores, os quais utilizam instruções normais para referenciamento de memória, diferente das configurações de multiprocessadores. É interessante destacar que, como cada nó é isolado um do outro, este tipo de estrutura se torna ideal para computação de alta-disponibilidade.

Como exemplos de configurações multicomputadores, podemos citar os arranjos de grids e clusters. Neste último, ao interconectar vários computadores comuns, seja por barramento ou equipamento comutador (*switch*), estes agem como um sistema único para o usuário (DANTAS, 2005).

## 2.2 Sistemas distribuídos

Desde os primórdios da era moderna da computação, em 1945, até aproximadamente 1985, os computadores eram grandes e caros, com minicomputadores custando na faixa de dezenas de milhares de dólares. Dessa forma, muitas empresas acabavam tendo poucos computadores, os quais, dada a falta de um método de interconexão, operavam individualmente (TANENBAUM; STEEN, 2002).

A situação começa a mudar em meados dos anos 1980, com avanços tecnológicos marcantes. Um desses avanços é o desenvolvimento de microprocessadores mais poderosos – partindo de 8 bit com uma rápida evolução para arquiteturas de 16, 32 e 64 bits – sendo alguns deles comparáveis em desempenho a um mainframe da época, com um custo muito menor. O progresso obtido foi tanto que se passou de máquinas que custavam 100 milhões de dólares e executavam 1 instrução por segundo para máquinas que custavam 1000 dólares e executavam 10 milhões de instruções por segundo, um ganho de performance versus preço de  $10^{12}$  (TANENBAUM; STEEN, 2002).

Segundo Tanenbaum e Steen (2002), o advento das redes de computadores de alta velocidade é outro grande avanço da época. As Redes de Área Local (LAN - *Local Area Network*) permitem a interconexão de centenas de computadores em um mesmo edifício, provendo a troca de pacotes de informações entre eles em poucos microssegundos. Não se limitando a somente um edifício, as Redes de Longa Distância (WAN - *Wide Area Network*), por sua vez, possibilitam a interconexão de computadores ao redor de todo o globo terrestre, podendo atingir taxas de transferências na escala de gigabits por segundo.

O constante crescimento do tamanho e complexidade dos problemas, bem como a necessidade cada vez maior destes serem resolvidos em menos tempo é um desafio constante. Dantas (2005) expõe três alternativas: utilizar um sistema com processador mais potente, otimizar o algoritmo e/ou trocá-lo por um mais eficiente ou, ainda, adotar o paradigma da computação distribuída ou paralela.

Analisando a primeira alternativa, é importante mencionar Gordon Moore, um dos fundadores da Intel, o qual predisse que o poder computacional dobraria a cada período de dezoito meses, de forma constante (MOORE, 1965). A previsão de Moore foi concretizada durante muitos anos, sendo até superada em alguns casos, mas limites como a velocidade da luz, leis de termodinâmica e crescente custo na produção de novos processadores vêm intrincando o avanço até então atingido (DANTAS, 2005).

Segundo Dantas (2005), a otimização do algoritmo ou a troca por um mais eficiente seria uma solução plausível em um primeiro momento, mas o alto custo para

otimização de um algoritmo existente se torna um impeditivo. Assim, muitas empresas optam por continuarem executando aplicações ineficientes, sem se preocuparem com o seu desempenho.

Levando em consideração os entraves das duas primeiras alternativas, uma opção que surge para o aumento do desempenho é o agrupamento de computadores, a fim de executarem a mesma aplicação de forma paralela ou distribuída. Estes computadores podem ser locais e interconectados por uma LAN, caracterizando um cluster, ou geograficamente distribuídos e interconectados por uma WAN, caracterizando um grid. Ambas as caracterizações fazem parte do que se define como sistemas distribuídos (DANTAS, 2005; TANENBAUM; STEEN, 2002).

### **2.2.1 Definição dos sistemas distribuídos**

Para Tanenbaum e Steen (2002), um sistema distribuído pode ser definido como um aglomerado de computadores independentes, os quais se mostram para o usuário como um sistema uniforme, sem que ele perceba que está utilizando mais do que uma máquina.

Segundo Kuz, Chakwarty e Heiser (2007), sistemas distribuídos atuais proveem comumente uma tarefa ou serviço único através da união de vários computadores independentes. Para Pereira Filho (2004), este tipo de sistema pode ser visto simplesmente como um aglomerado de processos simultâneos, os quais executam o mesmo trabalho cooperativamente, em vários nós distintos fracamente associados.

Na visão de Coulouris et al. (2012), componentes e aplicações que se encontram em computadores conectados em rede, capazes de se comunicarem e coordenarem suas ações através de simples trocas de mensagens, caracterizam um sistema distribuído. Sistema este que pode estar separado geograficamente por qualquer distância, seja no outro lado do mundo, seja na mesma sala.

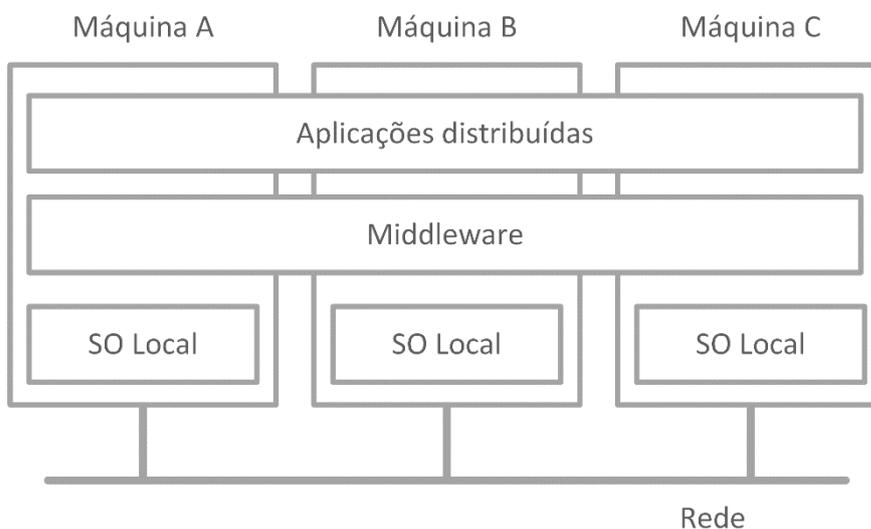
É importante observar no que tange aos sistemas distribuídos a necessidade de estes serem facilmente expandidos ou escalonados. Tal tipo de sistema deve apresentar uma disponibilidade constante, apesar de possíveis falhas ou

indisponibilidade em determinados nós. Trocas e/ou adição de novos componentes devem ocorrer de forma imperceptível ao usuário final do sistema (TANEMBAUM; STEEN, 2002).

Um sistema distribuído pode ser composto por computadores com componentes e sistemas exatamente iguais, configuração essa chamada de homogênea, ou por computadores com componentes e sistemas diferentes entre si, formando uma configuração heterogênea. Independentemente da configuração do sistema distribuído, este deve-se apresentar como algo único ao usuário final (DANTAS, 2005).

A fim de possibilitar essa percepção de sistema único para o usuário, Tanenbaum e Steen (2002) propõem uma organização de sistema distribuído com uma camada de *middleware*<sup>1</sup> se estendendo por todos os nós deste, como pode ser observado na Figura 3.

Figura 3 – Um sistema distribuído organizado como *middleware*



Fonte: Adaptada pelo autor, com base em Tanenbaum e Steen (2002).

<sup>1</sup> O *middleware* se caracteriza por um software específico, executado entre o sistema operacional e as aplicações, com o intuito de ocultar a essência heterogênea e distribuída do arranjo computacional. É responsável pela abstração da uniformidade do sistema computacional para o usuário final (BAKER, 2000).

## 2.2.2 Rede e interconexão

A fim de possibilitar a intercomunicação dos seus diversos nós, um sistema distribuído pode se utilizar de redes locais ou redes de longa distância. Coulouris et al. (2012) enfatizam os aspectos cada vez mais demandados em um sistema distribuído moderno: escalabilidade, mobilidade, confiabilidade e segurança.

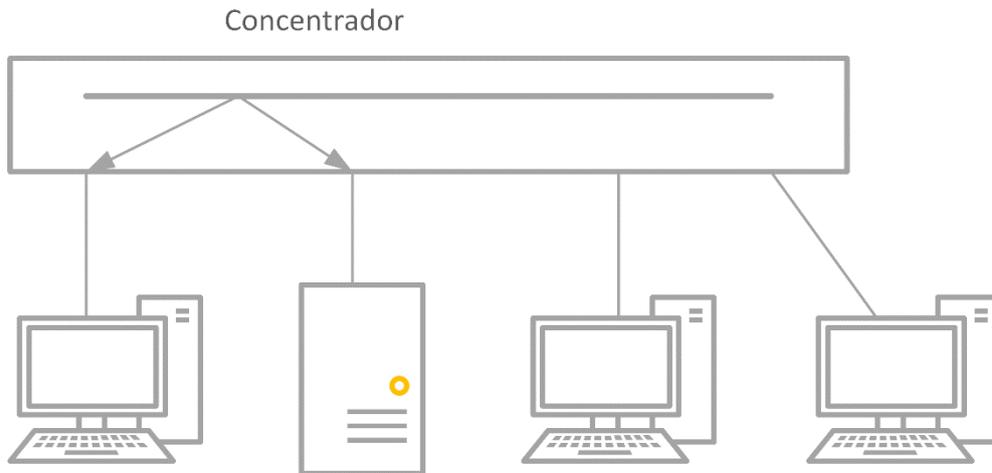
Dantas (2005) ressalta que, a fim de atender a esses aspectos, as redes de computadores vêm demonstrando constante evolução; entretanto, em alguns casos, não na mesma velocidade e proporção do desempenho das aplicações que as utilizam. Nesta seção, são abordados dispositivos de interconexão convencionais, bem como de alto desempenho.

### 2.2.2.1 Dispositivos convencionais de interconexão

Quanto aos dispositivos convencionais de interconexão de rede, destacar-se-ão, neste trabalho, os concentradores (*hubs*) e os comutadores (*switches*). Esses dois dispositivos se tornaram populares na década de 90, quando as redes do tipo *Ethernet* começaram a adotá-los, abandonando, gradativamente, a interligação coaxial (DANTAS, 2005).

Um *hub* pode ser definido como um dispositivo repetidor, atuando sob bits individuais, os quais, ao serem recebidos por uma interface do *hub*, são recriados, potencializados e transmitidos para todas as outras interfaces. Esse comportamento pode levar a colisões de quadros caso dois nós da rede tentem transmitir ao mesmo tempo, obrigando-os a retransmitir em momentos diferentes um do outro (KUROSE; ROSS, 2013).

Segundo Dantas (2005), este tipo de dispositivo atua comumente na faixa dos 10 e 100 Mbps (Megabits/segundo), fazendo uso de cabos de par trançado para interligação entre os computadores e ele. A Figura 4 demonstra, de forma simplificada, o funcionamento de um *hub*.

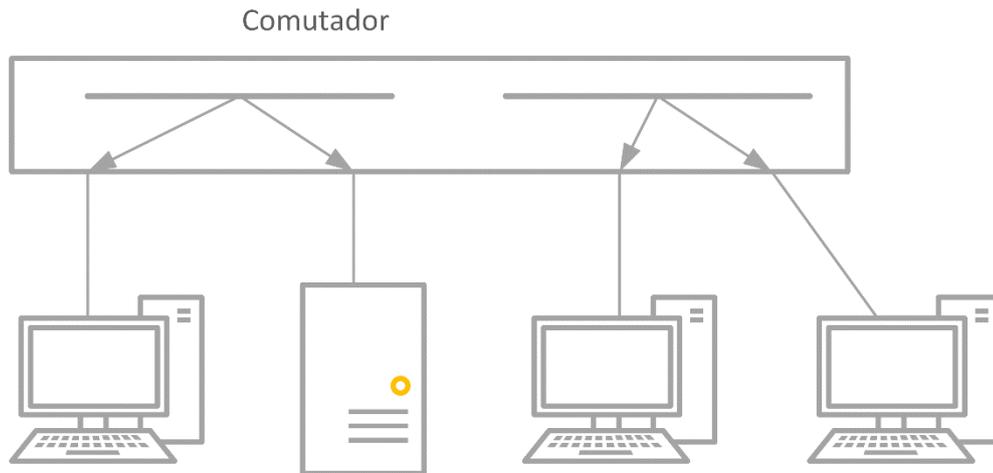
Figura 4 – Funcionamento simplificado de um *hub*

Fonte: Adaptada pelo autor, com base em Dantas (2005).

Já dispositivos do tipo *switch* atuam como comutadores, permitindo múltiplas conexões paralelas, sendo necessário apenas que os computadores requestem suas ligações em pares. Este tipo de dispositivo pode atuar na faixa dos Gigabits/segundo (DANTAS, 2005).

Kurose e Ross (2013) destacam que este tipo de equipamento nunca transmite mais de um quadro ao mesmo tempo em uma mesma interface. Também ressaltam que *switches* modernos atuam em *full-duplex*, possibilitando uma comunicação de duas vias entre o *switch* e o nó, sem interferências ou colisões. A Figura 5 demonstra o funcionamento simplificado de uma configuração comutada, em que é representada a ligação das máquinas A e B paralelamente às máquinas C e D.

Figura 5 – Funcionamento simplificado de um *switch*



Fonte: Adaptada pelo autor, com base em Dantas (2005).

É interessante destacar que, com a utilização de *switches* corretamente mensurados, é possível construir um cluster de computadores que ultrapasse a marca de mil nós (STERLING, 2002).

### 2.2.2.2 Dispositivos de interconexão de alto desempenho

Em arranjos de clusters e grids computacionais de alto desempenho, os dispositivos de interconexão devem contar com ampla escalabilidade e tolerância a falhas, bem como altas taxas de transmissão aliadas à baixa latência. Dessa forma, *switches* comuns, mesmo que na classe Gigabit Ethernet, podem não ser suficientes para suprir a demanda do sistema como um todo (DANTAS, 2005).

Com estes requisitos em mente, surgem várias tecnologias de interconexão para sistemas distribuídos de alto desempenho, as quais Dantas (2005) divide em duas classes. A primeira abrange equipamentos que fazem uso de um meio fundamentado pela programação de transferência de mensagens entre processadores na camada da placa de rede, sendo destaques as tecnologias Myrinet, Gigabyte System Network, GigaNet e InfiniBand. Já a segunda classe caracteriza-se por gerar uma concepção de memória virtual única para todos os computadores interconectados no equipamento. São exemplos desta última classe o Dolphin SCI e o Quadrics Network (QSNET).

De acordo com o Scogland, Subramaniam e Feng (2012), em 2011, a InfiniBand foi a tecnologia de interconexão mais utilizada entre os 30 computadores “mais verdes” do ranking Green500, conforme a Tabela 1.

Tabela 1 – Tecnologias de interconexão nos 30 computadores “mais verdes” da Green500 com o passar dos anos

	2007	2008	2009	2010	2011
<b>Proprietária/ Customizada</b>	26	12	18	16	5
<b>InfiniBand</b>	4	18	12	13	25
<b>Gig-E</b>	0	0	0	1	0
<b>% customizada (toda a lista)</b>	13%	12%	9%	10%	12%

Fonte: Scogland, Subramaniam e Feng (2012, p. 4).

### 2.3 Clusters

Segundo Dantas (2005), os arranjos de clusters aliam a capacidade de prover melhor e maior poder computacional a um ótimo custo-benefício, levando-os a serem adotados por diversas instituições. Estes arranjos vêm sendo usados para renderização gráfica (HUMPHREYS et al., 2002), sensoriamento remoto (YANG et al., 2001), astrofísica (DUBINSKI et al., 2003), entre outros.

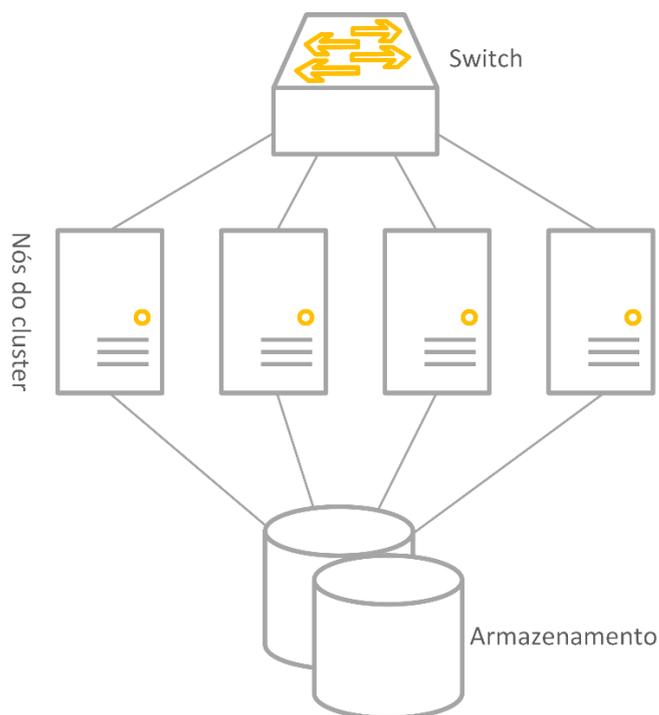
Outros fatores que contribuem para essa adoção são a sua grande escalabilidade e facilidade de configuração aliadas à grande variedade de computadores ofertados no mercado, desde os mais comuns de baixo custo até os mais especializados e custo mais elevado (DANTAS, 2005). Soluções como o GNU/Linux, uma plataforma de software livre e que oferece distribuições especializadas em clusters, ajudam a difundir ainda mais este paradigma (KOPP, 2012).

Pode-se definir um cluster como um aglomerado de computadores, os quais operam de forma conjunta para a realização de uma tarefa computacional específica.

Um cluster é formado por diversos computadores, chamados de nós (podendo também ser referenciados como *nodes* ou *nodos*). Como qualquer outro arranjo de sistema distribuído, o cluster deve ser percebido pelo usuário como um sistema uniforme (PEREIRA FILHO, 2004; KOPP, 2012).

Por se tratar de uma arquitetura multicomputador (vide 2.1.2), Pereira Filho (2004) aponta que os clusters são fracamente agrupados, necessitando, dessa forma, de uma comunicação entre os nós, a fim de possibilitar a correta distribuição das instruções a serem exercidas. Apesar de não dividirem o mesmo barramento, alguns outros elementos de hardware devem ser compartilhados a fim de prover uma tolerância a falhas aceitável. Na Figura 6, pode-se observar um cluster com quatro nós, compartilhando o hardware de rede e armazenamento.

Figura 6 – Cluster de quatro nós compartilhando o hardware de rede e armazenamento



Fonte: Adaptada pelo autor, com base em Pereira Filho (2004).

### 2.3.1 Objetivos de um cluster

De acordo com Vogels et al. (1998), um cluster deve atender aos seguintes objetivos para ser considerado adequado:

- I) **Comodidade:** o cluster deve ser executado em um conjunto de computadores comuns, rodando um sistema operacional uniforme. Eles devem ser interligados por uma rede genérica, na qual a comunicação deve ocorrer de acordo com protocolos padrões da internet.
- II) **Escalabilidade:** a adição de aplicações, nós, periféricos e interconexões de rede deve ocorrer de forma que não interrompa a disponibilidade dos serviços oferecidos pelo cluster.
- III) **Transparência:** apesar de ser formado por diversos computadores, o cluster deve se apresentar ao usuário como um sistema único. As aplicações do cliente e as ferramentas de gerenciamento enxergam o cluster como se fosse um grande servidor de alto desempenho e confiabilidade.
- IV) **Confiabilidade:** o cluster deve ter ferramentas capazes de detectar possíveis falhas em seus componentes de hardware e software, agindo também de forma a evitar que estas comprometam os serviços disponibilizados por ele.

### 2.3.2 Abstrações de um cluster

Segundo Pereira Filho (2004), um cluster é formado por diversos elementos, que, quando unidos, proveem a funcionalidade almejada. Nas subseções seguintes, são abordados tais elementos de forma abstraída.

#### 2.3.2.1 Nó

O nó é o elemento básico do cluster, caracterizado por um sistema computacional, o qual deve ser capaz de se comunicar com outros nós do grupo,

através de mensagens via rede. O nó deve ser composto por quatro elementos essenciais, que são: CPU, memória, armazenamento e canal de interconexão.

### **2.3.2.2 Recurso**

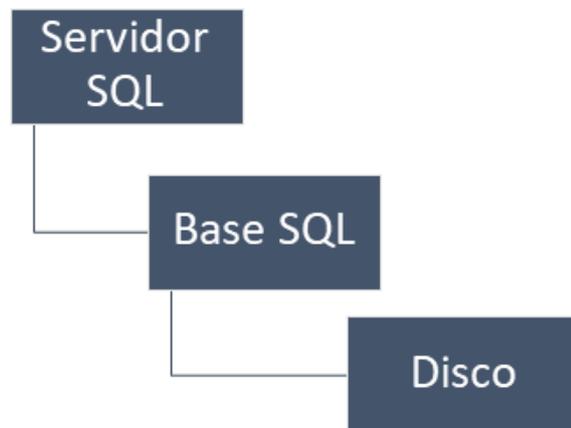
Um recurso pode ser visto como uma funcionalidade disponibilizada por um nó. Este pode ser lógico, com um endereço de IP, ou físico, como uma câmera ou impressora. Os recursos devem ser facilmente migráveis entre um nó e outro, atendendo aos requisitos de disponibilidade em caso de falhas.

Cabe ao sistema gerenciador do cluster monitorar o estado dos serviços associados a esses recursos, tomando ações de início e parada em caso de algum evento adverso.

### **2.3.2.3 Dependência de recursos**

Em muitos casos, determinados recursos dependem da disponibilidade de outros recursos do cluster para o seu correto funcionamento. Tomando de exemplo um servidor *Structured Query Language* (SQL), este depende da presença de uma determinada base SQL, que, por sua vez, depende da disponibilidade dos discos que armazenam a base. Conforme a Figura 7, gera-se uma árvore de dependências, a qual deve ser utilizada caso ocorra a migração de recursos, indicando quais devem ser migrados de forma conjunta, bem como a ordem correta que estes devem ser iniciados (VOGELS, 1998).

Figura 7 – Árvore de dependências de um servidor SQL



Fonte: Elaborado pelo autor, com base em Pereira Filho (2004) e Vogels (1998).

#### 2.3.2.4 Grupo de recursos

Um grupo de recursos caracteriza-se pelo agrupamento de recursos, os quais devem ser migrados de forma conjunta, garantindo o correto funcionamento de cada um deles. O administrador do sistema pode agrupar recursos independentes, mesmo que estes, apesar de necessários, não constem na árvore de dependências.

É importante observar dois pontos: a migração de recursos deve sempre obedecer ao grupo de recursos, e uma árvore de dependências nunca deve exceder os limites de um grupo.

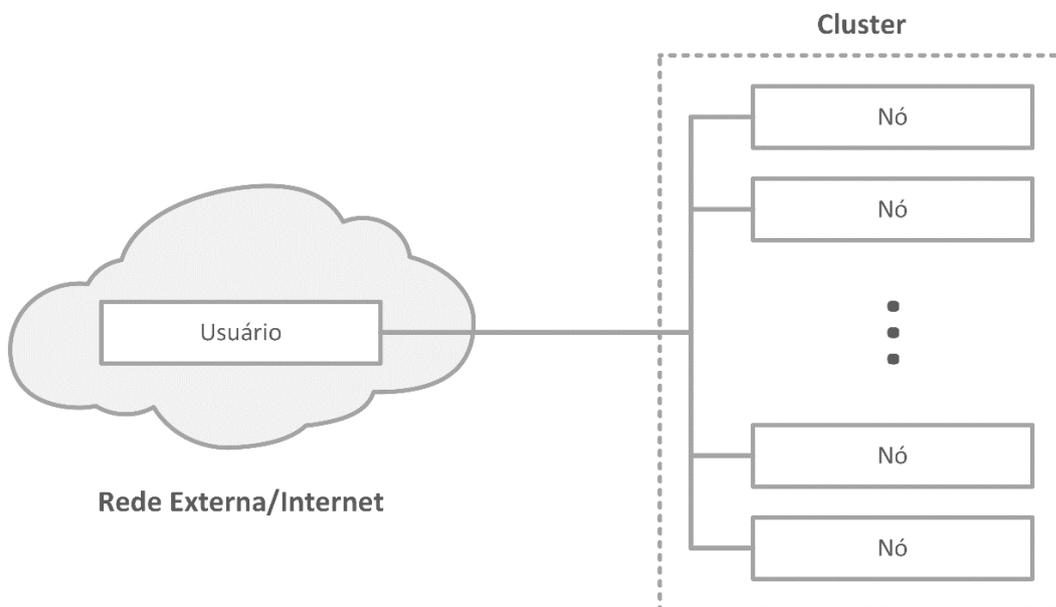
#### 2.3.3 Estrutura de um cluster

Segundo Sloan (2004), não podemos assumir que um cluster é somente um punhado de computadores interconectados de forma aleatória. É necessária atenção à estrutura destes no momento de sua construção, definindo qual função cada nó assumirá, bem como a maneira que a rede de interconexão será organizada.

### 2.3.3.1 Cluster simétrico

A estrutura mais simplificada de cluster é o simétrico, o qual pode ser facilmente estruturado através da criação de uma sub-rede de computadores ou, até mesmo, utilizando computadores de uma rede local. Estes, por sua vez, serão gerenciados por um software gerenciador de cluster, a fim de executarem as tarefas desejadas (SLOAN, 2004). Na Figura 8, podemos ver uma representação de um cluster simétrico.

Figura 8 – Cluster simétrico



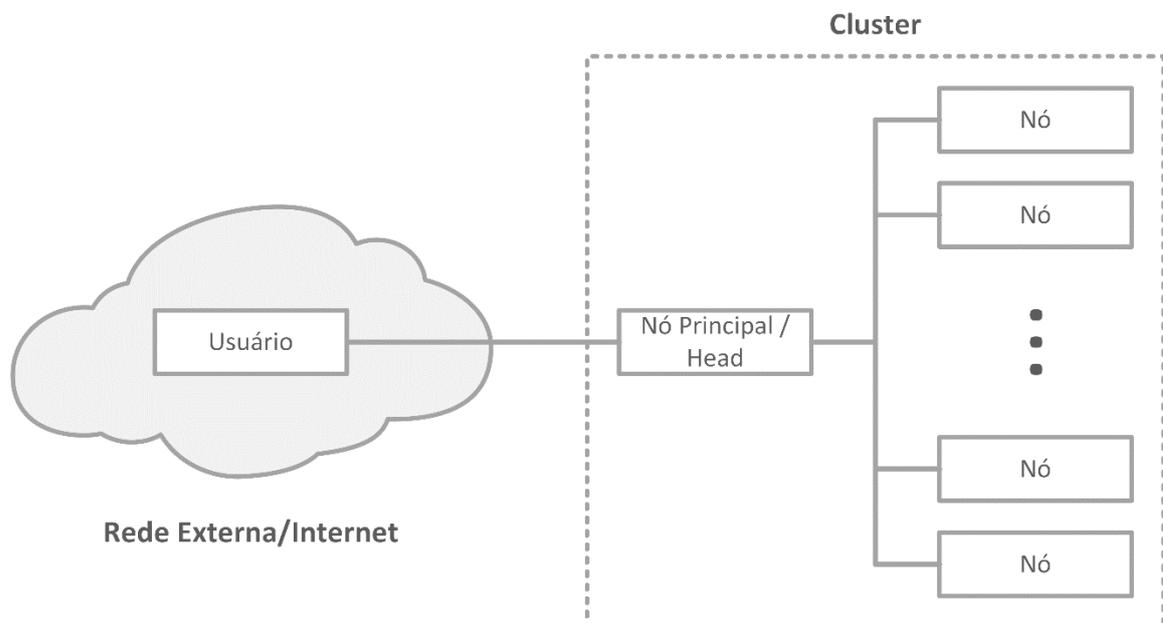
Fonte: Adaptada pelo autor, com base em Sloan (2004).

Sloan (2004) destaca que, neste tipo de arranjo, todos os computadores podem também ser utilizados individualmente quando não estiverem sendo demandados pelo cluster. Dada essa característica e a ausência de um controle central, este tipo de abordagem de clusters não se mostra muito seguro, nem muito prático de ser gerenciado.

### 2.3.3.2 Cluster assimétrico

Clusters com arquitetura do tipo assimétrica são mais comuns em configurações dedicadas. Neste tipo de arranjo, um nó é configurado e otimizado para agir como o nó principal ou *head-node*, atuando como uma camada extra entre os demais nós do cluster e os usuários finais do sistema, conforme a Figura 9 (SLOAN, 2004).

Figura 9 – Cluster assimétrico



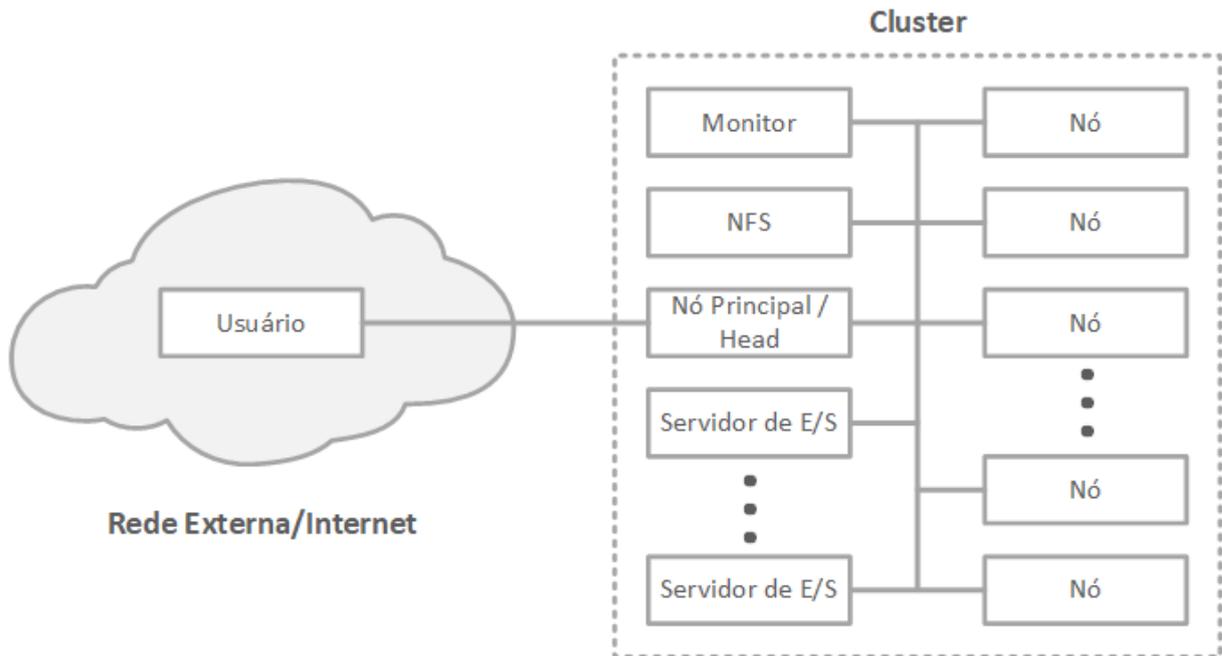
Fonte: Adaptada pelo autor, com base em Sloan (2004).

Todo o tráfego do cluster irá passar pelo nó principal, tornando este tipo de configuração muito mais segura e fácil de ser gerenciada. Comumente, o nó principal possui especificações superiores de hardware e software para lidar com esse tráfego, enquanto os demais nós computacionais operam com configurações de software mínimas, exclusivas para o seu funcionamento dentro do cluster (SLOAN, 2004).

Em clusters maiores, Sloan (2004) destaca que só aumentar a capacidade do nó principal pode não ser suficiente. Uma alternativa para essa limitação é a especialização de mais nós no mesmo nível do principal, como, por exemplo, configurar um nó somente para monitoramento das condições do cluster, outro para atuar como *Network File Server* (NFS - Servidor de Arquivos da Rede), bem como

outro para lidar com o tráfego de entrada e saída (E/S). Um cluster desse tipo é referido pelo autor como um cluster expandido, e sua representação pode ser vista na Figura 10.

Figura 10 – Representação de um cluster expandido com nós especializados para monitoramento, servidor NFS e servidor de E/S



Fonte: Adaptada pelo autor, com base em Sloan (2004).

#### 2.3.4 Aplicações-alvo

De acordo com Kopp (2012), pode-se definir um cluster de acordo com sua função específica em um dado cenário, de modo que pode ser classificado como: de alto desempenho, de alta disponibilidade e de balanceamento de carga.

Dantas (2005) destaca que, apesar de existirem essas classificações, em alguns cenários é possível observar mais do que um tipo de requisito, formando, por exemplo, um cluster de alto desempenho e alta disponibilidade. Cada classificação será detalhada nas subseções subsequentes.

#### 2.3.4.1 Cluster de alto desempenho

Um cluster de alto desempenho tem por finalidade sustentar aplicações que demandem um alto poder de processamento para sua execução. O ambiente com diversos processadores e grandes quantidades de memória e de armazenamento tornam este tipo de cluster ideal para isso (DANTAS, 2005).

Podendo também ser referido como Cluster *High Performance Computing* (HPC), este tipo de cluster se especializa em dividir as tarefas entre os seus diversos nós interconectados, de forma a realizá-la de forma mais eficiente, entregando um resultado em menor tempo (PEREIRA FILHO, 2004).

De acordo com Kopp (2012), clusters deste tipo são comumente utilizados para computação científica, análises financeiras, dentre outras aplicações que demandem alto poder de processamento.

#### 2.3.4.2 Cluster de alta disponibilidade

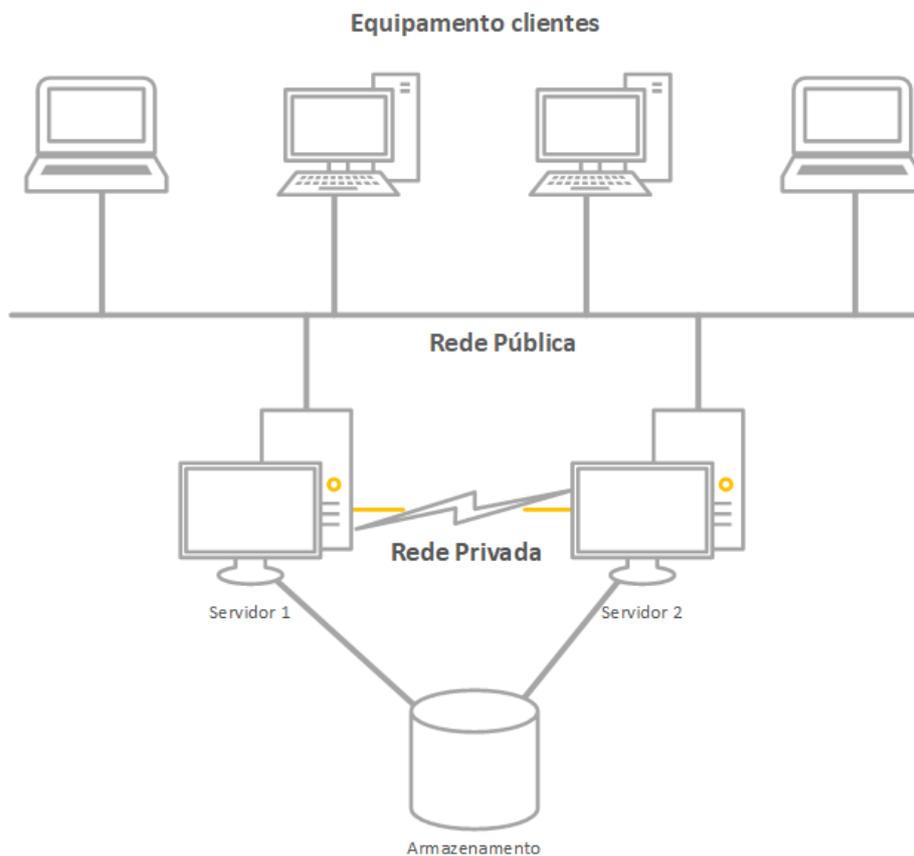
Também conhecidos como clusters de *failover* ou *High Availability* (HA), este tipo de cluster tem como objetivo principal sustentar serviços e aplicações críticas, as quais não podem ficar indisponíveis. A redundância é o elemento essencial, dado que várias máquinas idênticas, tanto em hardware quanto em software, compõem a estrutura (KOPP, 2012).

Ainda segundo Kopp (2012), em um cluster de *failover*, é comum que somente a máquina primária esteja disponível de forma direta enquanto as outras permanecem em um estado latente. Caso a máquina principal venha a falhar, uma das máquinas secundárias que estava latente e monitorando a primária assume o papel principal, garantindo a disponibilidade do serviço prestado.

Pitanga (2008) destaca que este tipo de cluster pode atingir um nível de disponibilidade de 99,99%, o que é muito superior se comparado a um servidor de boa qualidade, que demonstra uma disponibilidade de 99,5%. Estes índices se tornam atrativos para diversas empresas, as quais dependem essencialmente de seus

sistemas para se manterem no mercado, como, por exemplo, um site de comércio eletrônico. Na Figura 11, pode-se observar a representação de um cluster de alta disponibilidade com dois nós.

Figura 11 – Cluster de alta disponibilidade com dois nós



Fonte: Elaborada pelo autor, com base em Pitanga (2008).

### 2.3.4.3 Cluster de balanceamento de carga

Clusters deste tipo tem como foco proporcionar melhor desempenho a aplicações como servidores web (KOPP, 2012) ou ambientes virtualizados, os quais disponibilizam diversas máquinas virtuais com sistemas operacionais completos para os usuários finais (KUDRYAVTSEV; KOSHELEV; AVESTISYAN, 2012). É importante destacar que clusters deste tipo comumente estão associados aos clusters de alta disponibilidade, tanto que alguns autores nem os tratam como uma classificação separada.

De acordo com Kopp (2012), eles atingem seu objetivo distribuindo a carga de trabalho entre os diversos nós do cluster, através de algoritmos como *Round-Robin DNS (Domain Name System)* em um servidor web. Nesse caso, ocorre uma espécie de rodízio, em que, a cada consulta DNS de um IP (*Internet Protocol*) diferente, o algoritmo retorna o endereço da próxima máquina do cluster. Em muitos casos, um algoritmo como o *Round-Robin DNS* não se mostra muito eficiente, sendo que algoritmos mais avançados se baseiam no retorno do estado de cada nó do cluster, decidindo, assim, para qual deles irá distribuir a tarefa de acordo com sua carga.

### **2.3.5 Tipos de computadores empregados**

Pode-se, ainda, classificar os clusters quanto aos tipos de computadores empregados em sua estrutura.

#### **2.3.5.1 NOW / COW**

Segundo Sloan (2004), os clusters do tipo NOW (*Network of Workstations*) ou COW (*Cluster of Workstations*) empregam computadores que são utilizáveis individualmente como estações de trabalho quando não atuantes no cluster. Pitanga (2008) destaca que, neste tipo de arquitetura, diferentemente do que ocorre no Beowulf, é possível ter acesso ao teclado (console) em qualquer nó da rede.

Estes computadores podem estar distribuídos, por exemplo, numa rede local de um escritório ou de um laboratório de uma universidade. Em momentos de ociosidade, estes computadores atuam como nós de um cluster (SLOAN, 2004).

#### **2.3.5.2 Beowulf**

Compartilhando muitas características com os clusters do tipo COW, os clusters Beowulf também se caracterizam por serem compostos por computadores comuns, com peças facilmente encontradas no mercado. Uma diferença diz respeito à estrutura, a qual, neste caso, é assimétrica, com um nó principal comandando todos

os outros nós, chamados aqui de escravos. Os nós escravos não são utilizáveis fora do cluster, sendo dedicados somente a este, não contando com teclado, mouse ou tela (PITANGA, 2008).

Este tipo de cluster originou-se de um projeto da Agência Espacial Norte Americana (NASA - *North American Space Agency*), em 1994, quando um grupo de cientistas necessitava de um supercomputador capaz de processar na ordem de um gigaflop, a fim de atender às demandas de simulações complexas. Devido aos altos custos de um sistema com tal potência, os pesquisadores Tomas Sterling e Don Becker agregaram 16 computadores pessoais, compostos por processadores Intel 486, sistema operacional Linux e rede Ethernet, os quais, em conjunto, atingiram a marca de 70 megaflops a um custo de apenas 50 mil dólares (PITANGA, 2008; SLOAN, 2004).

Clusters do tipo Beowulf são caracterizados por atingirem alto desempenho a um baixo custo: entre um décimo a um centésimo do valor de um supercomputador especializado equivalente. Esta característica foi e continua sendo um grande atrativo para a adoção deste tipo de arranjo por parte da comunidade científica e técnica, aliado ao uso de software livre, ou, até para usuários comerciais, que acabam adotando soluções de software especializado pago, como o Microsoft Windows (STERLING; LUSK, 2002; MCGARVEY et al., 2001).

### **2.3.5.3 CLUMPS**

Um *Cluster of SMPs* (CLUMPS) caracteriza-se por um agregado de computadores com a arquitetura *Symmetric Multi-Processing* (SMP - Multiprocessamento Simétrico) (DANTAS, 2005). Este tipo de computador se caracteriza por adotar o paradigma de multiprocessador, detalhado anteriormente neste trabalho, em que um único computador conta com diversos processadores, com memória compartilhada entre eles.

Ao agregar diversas máquinas SMPs em um cluster, é possível dispor de mais processadores, empregando um menor número de máquinas. Conseqüentemente, o

sistema ficará mais fácil de ser gerenciado, e o espaço físico ocupado será menor (PITANGA, 2008).

### **2.3.6 Tipos de nós**

Um cluster também é classificado quanto a semelhança entre os softwares e componentes de hardware de seus nós, podendo ser denominado como homogêneo ou heterogêneo (DANTAS, 2005).

#### **2.3.6.1 Cluster homogêneo**

É dito que um cluster é homogêneo quando todos os seus nós possuem a mesma configuração de hardware e software, além da mesma rede de interconexão (PITANGA, 2008).

Em um ambiente homogêneo, tarefas de distribuição e balanceamento de carga tornam-se mais eficientes e facilitadas, pois, como possuem configurações iguais, todos recebem a mesma carga de trabalho. A preocupação com interoperabilidade de software torna-se nula; e a reposição de componentes, facilitada, visto a igualdade de todos os nós (DANTAS, 2005; PITANGA, 2008).

De acordo com Pitanga (2008), apesar de apresentarem a configuração ideal, muitos clusters homogêneos passam a ser heterogêneos com o tempo. Isso ocorre, principalmente, quando há necessidade de ampliação do cluster e/ou quando algum de seus nós sofre algum defeito e precisa ser substituído. Em ambas as situações, quase sempre será um computador mais moderno que assumirá a posição no cluster.

#### **2.3.6.2 Cluster heterogêneo**

Um cluster heterogêneo é formado por computadores com configurações de hardware diferentes entre si ou, até mesmo, que utilizam diferentes redes de intercomunicação entre determinados grupos de computadores. Este tipo de cluster é

comumente encontrado em projetos que cresceram de forma modular, em que novos nós foram sendo adicionados ao arranjo com o passar do tempo (PITANGA, 2008; DANTAS, 2005).

Por se tratar de computadores com configurações diferentes entre si, o balanceamento de carga se torna mais complexo e deve ser feito com maior detalhe. Se efetuado corretamente, o balanceamento irá evitar que nós com configurações inferiores fiquem sobrecarregados e nós mais eficientes fiquem ociosos (PITANGA, 2008).

### **2.3.7 Categorias dos clusters**

De acordo com Dantas (2005), grande parte dos supercomputadores atuais, que figuram na lista TOP500, adotam arquiteturas baseadas em clusters. Grande parte desses sistemas são compostos por peças encontradas facilmente no mercado, as chamadas “mercadorias de prateleira” ou COTS (*Commodity Off the Shelf*), mas utilizando soluções de interconexão de alto desempenho, como a InfiniBand.

Dado este tipo de especialização, Dantas (2005) divide os clusters em duas categorias:

- Categoria I: esta categoria abrange os clusters que fazem uso de peças e softwares personalizados para o arranjo. Mesmo com determinado nível de personalização, esta categoria de clusters também faz uso de peças COTS, mas de uma maneira industrial. Como exemplo, pode-se citar o multicomputador Altix 3700, da Silicon Graphics (SGI), o qual pode utilizar de 16 a 256 processadores Intel Itanium2, um componente COTS, por nó. Ele também adota uma arquitetura de memória compartilhada, podendo atingir de 4 a 8 TB (Terabyte) por nó.
- Categoria II: já esta categoria adota exclusivamente componentes COTS e conjuntos de softwares livres, arranjados de forma “artesanal” ou “*self-made*”. Fazem uso de interfaces como a Ethernet para sua interconexão e barramentos SCSI (*Small Computer System Interface*) e SATA (*Serial Advanced Technology Attachment*) para suas interfaces de entrada e saída.

### 2.3.8 Software e gerenciamento

Gerenciar um cluster é um trabalho que envolve vários fatores, incluindo a instalação do sistema operacional e etapas como a definição das ferramentas de configuração, manutenção, monitoramento e distribuição de tarefas (SCHEPKE et al., 2005).

Visto a quantidade de computadores presentes em um cluster e a necessidade de instalar o mesmo sistema operacional e softwares em cada um dos nós, ferramentas de automação são comumente empregadas para este tipo de configuração. O *Kickstart* para o Red Hat Linux e o *Open Source Cluster Application Resources* (OSCAR) são exemplos de utilitários que facilitam a configuração inicial de um cluster, praticamente eliminando qualquer necessidade de interação do usuário para essa tarefa (SCHEPKE et al., 2005).

O emprego de um sistema de imagem única (SSI - *Single System Image*) também é outro ponto importante a ser observado no momento de configuração do cluster. Dantas (2005) explica que um sistema deste tipo pode atuar na camada de *middleware*, do sistema operacional ou, até mesmo, em nível de hardware. Seja qual for o nível aplicado, o objetivo do SSI é tornar o cluster um ambiente único para o usuário, como se fosse somente um sistema, facilitando seu uso e gerenciamento.

Segundo Schepke et al. (2005), outro fator essencial no gerenciamento de um cluster é manter uma monitoração ativa deste. Diversas ferramentas, como o Parmon (BUYA, 2000) e o Pacemaker (LEVINE, 2018), possibilitam o monitoramento em tempo real dos recursos do cluster, como carga dos processadores, uso de memória e tráfego de rede. Estes dados são cruciais para o administrador do sistema, o qual pode gerenciar melhor a distribuição de tarefas, verificar nós defeituosos e até avaliar a necessidade de expansão do cluster, com a adição de novos nós.

## 2.4 Grids

O conceito de grid computacional surge em meados dos anos 1990, motivado pela crescente necessidade de poder computacional e os altos custos de soluções de

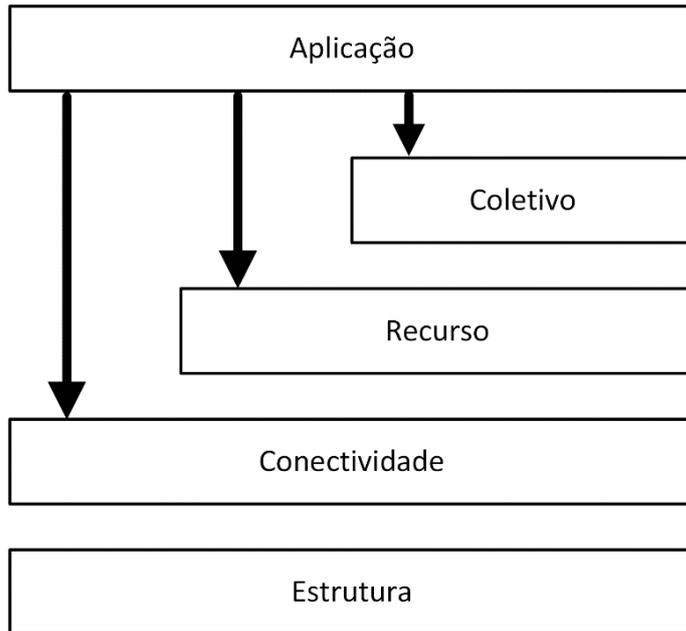
supercomputadores e clusters dedicados que pudessem atender a essa demanda. A ideia de interconectar, através da internet, máquinas já existentes de diferentes instituições geograficamente distribuídas, a fim de trabalharem na mesma tarefa, possibilitou um aumento expressivo no poder computacional a um custo muito menor e é o que define uma grid computacional (FOSTER et al., 2008).

Dantas (2005) afirma que uma grid computacional é um arranjo heterogêneo de computadores distribuídos geograficamente, o qual é acessível ao usuário final via uma interface única. Estes arranjos proveem uma quantidade muito maior de recursos e serviços compartilhados, quando comparados a outros tipos de configurações, vista a grande heterogeneidade de seus participantes. O acesso a esses serviços e recursos deve ocorrer de forma transparente ao usuário, sem necessidade deste saber onde os recursos estão fisicamente localizados.

#### **2.4.1 Arquitetura**

A fim de prover uma estrutura uniforme para descoberta e compartilhamento de recursos, as chamadas Organizações Virtuais (OV) – entidades lógicas que agregam os recursos como se estes fossem partes de uma mesma organização – as grids computacionais definem e proveem diversos protocolos e serviços essenciais. Estes protocolos e serviços são providos em cinco níveis diferentes, como representado na Figura 12 (FOSTER et al., 2008).

Figura 12 – Arquitetura protocolar de uma grid



Fonte: Adaptada pelo autor, com base em Foster et al. (2008).

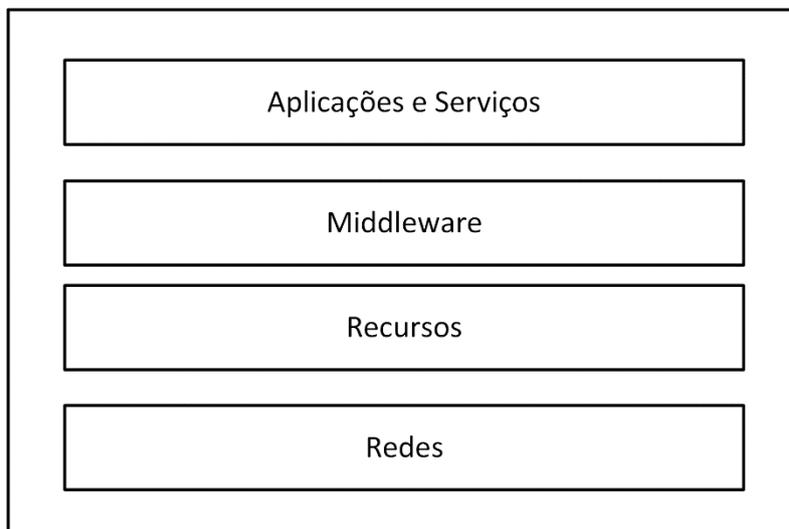
A seguir são descritas as características de cada uma das cinco camadas propostas por Foster et al. (2008):

- **Estrutura:** nesta camada, a grid fornece acesso a diversos recursos, como computação, armazenamento e rede. A grid comumente se baseia em componentes já existentes na estrutura, a fim de gerenciar os seus recursos e monitorar a qualidade de serviço (QoS - *Quality of Service*).
- **Conectividade:** esta camada define os protocolos de comunicação e autenticação essenciais à grid, possibilitando transações de rede mais seguras e facilitadas.
- **Recurso:** define protocolos que garantem segurança na publicação, descobrimento, negociação, monitoramento, controle, geração de relatórios e outros itens referentes às operações com recursos individuais. O *Grid Resource Access and Management* (GRAM) é um exemplo de protocolo usado a fim de alocar recursos computacionais e para monitorar e controlar o processamento nesses recursos.

- **Coletivo:** esta camada lida com as interações através de conjuntos de recursos, os quais são capazes de prover serviços como servidores de autorização e serviços de diretório (DANTAS, 2005).
- **Aplicação:** esta camada compreende toda e qualquer aplicação do usuário, as quais operam no ambiente da OV, construídas com base em protocolos e *Application Programming Interfaces* (APIs) das camadas inferiores.

Dantas (2005) apresenta uma segunda arquitetura proposta por Tony (2003), cuja composição se dá por meio de quatro camadas dispostas conforme a Figura 13.

Figura 13 – Arquitetura de grid proposta por Tony



Fonte: Elaborada pelo autor, com base em Dantas (2005).

De acordo com Dantas (2005), a seguir são explicadas as camadas do modelo a fim de melhor compreendê-lo:

- **Aplicações e Serviços:** um número expressivo de aplicações e serviços formam a camada superior de uma grid computacional. São incluídas aplicações diversas, abrangendo ferramentas de desenvolvimento e aplicações específicas de cada organização. Dentre os serviços, são abrangidas funções de gerenciamento chaves para correta provisão de recursos compartilhados entre os usuários.

- **Middleware:** camada caracterizada por fornecer protocolos a fim de possibilitar que elementos como servidores, redes e armazenamento façam parte de um âmbito de grid unificado. Deve contar, também, com protocolos a fim de suportar a heterogeneidade de hardware e software da grid.
- **Recursos:** camada formada pelos vários recursos que compõem a grid, como servidores e dispositivos de armazenamento.
- **Redes:** esta camada abrange toda a infraestrutura de conectividade dos recursos da grid, incluindo switches, roteadores e protocolos pertinentes.

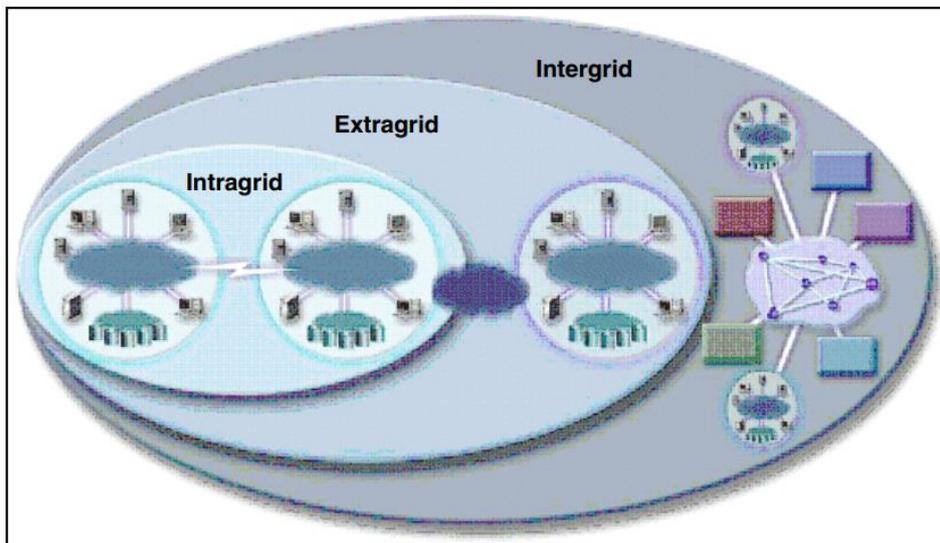
#### 2.4.2 Topologia

De acordo com Zhu e Ni (2003), pode-se classificar as grids em três categorias, de acordo com a topologia de interligação existente entre os seus elementos. Essas categorias são descritas a seguir e podem ser observadas na representação da Figura 14.

- **IntraGrid:** este tipo de topologia se caracteriza por uma grid que existe somente dentro de uma única organização. Neste caso, os vários elementos da grid compartilham um mesmo domínio administrativo e contam com largura de banda garantida na rede privativa que a interconecta. Essas características garantem um gerenciamento mais facilitado, bem como um nível de segurança mais elevado.
- **ExtraGrid:** uma extragrid é formada pela união de duas ou mais intragrids, comumente envolvendo mais que um domínio administrativo. Topologias desse tipo contam com segurança dispersa, dois ou mais domínios de segurança e conectividade via WAN. A dinamicidade dos recursos oferecidos também cresce. Por exemplo, duas instituições podem firmar uma parceria, unindo suas intragrids a fim de formar uma extragrid, compartilhando recursos entre si, os quais antes eram exclusivos e inacessíveis.
- **InterGrid:** análogo à internet, este tipo de topologia é constituído por centenas (ou até milhares) de grids computacionais de várias instituições. É o tipo mais

complexo de topologia de grid, contando com segurança dispersa, múltiplos domínios e conectividade WAN.

Figura 14 – Representação das topologias de intragrid, extragrid e intergrid



Fonte: Jacob et al. (2005, p. 104).

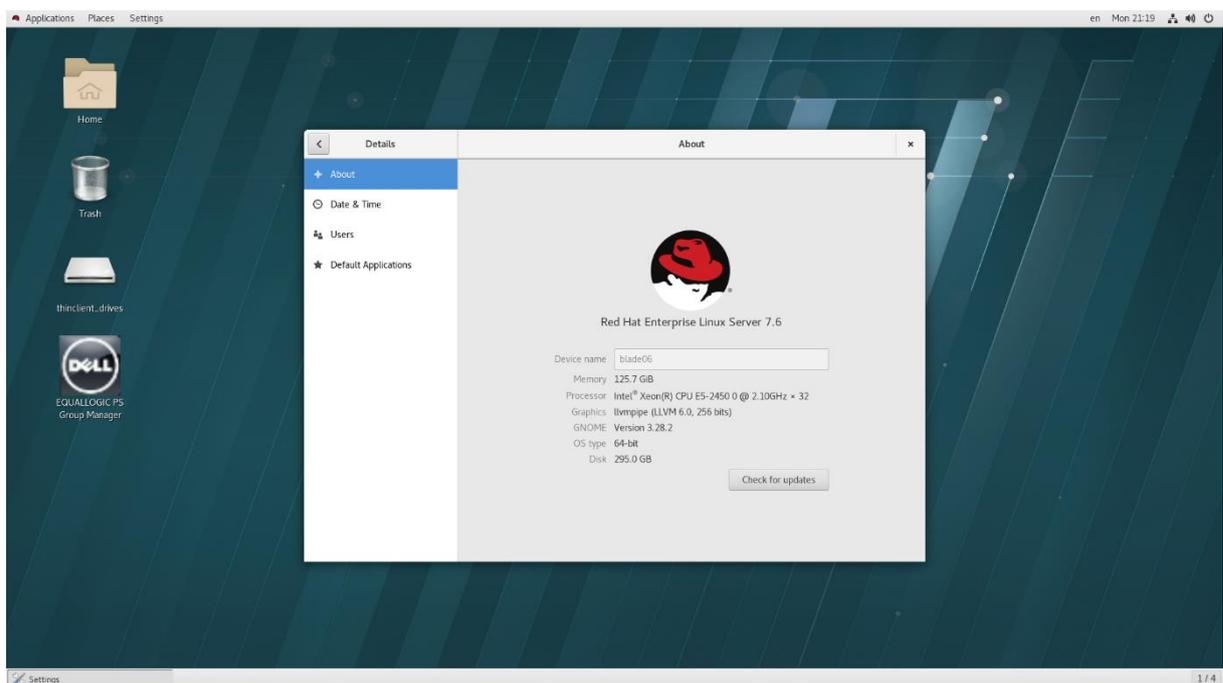
## 2.5 Red Hat Enterprise Linux

O sistema operacional Red Hat Enterprise Linux (RHEL) é uma distribuição do Linux, comercializada e mantida pela Red Hat, Inc. Ele recebeu este nome em 2003, quando a companhia dividiu sua famosa distribuição Red Hat Linux em duas: o Fedora Project, um sistema patrocinado pela companhia, mas sem suporte ou distribuição física, sendo totalmente *open-source*; e o RHEL, a versão comercial totalmente suportada pela companhia, voltada majoritariamente para empresas (NEGUS, 2009).

A Red Hat, Inc. comercializa o RHEL através do formato de assinaturas, por meio das quais fornece versões estáveis e testadas em intervalos regulares, atualizações de segurança e suporte por meio de canais diversos. A assinatura também disponibiliza acesso a manuais oficiais do sistema, fóruns exclusivos de suporte, treinamentos, programas de certificação e acesso aos repositórios de pacotes de software da Red Hat (NEGUS, 2009).

Ainda segundo Negus (2009), dada a sua constante evolução, alta escalabilidade e abrangente suporte, o RHEL tornou-se uma das mais respeitadas distribuições Linux para ambientes de alto desempenho, sendo adotada por grandes corporações, institutos de pesquisa e órgãos governamentais. A Figura 15 abaixo exhibe a área de trabalho gráfica do RHEL 7.6, a qual utiliza o *GNU Network Object Model Environment*<sup>2</sup> (GNOME), com a janela “Sobre” aberta.

Figura 15 – Interface gráfica do RHEL com a janela "Sobre" aberta



Fonte: Do autor (2019).

## 2.6 Virtualização

Um dos objetivos de um sistema operacional, de forma simplificada, é prover uma interface amigável entre o hardware no qual este está instalado e o usuário final. Esta interface provê ao usuário a capacidade de acessar, gerenciar e usufruir dos recursos presentes na máquina, através de aplicações e tarefas específicas (CARISSIMI, 2008).

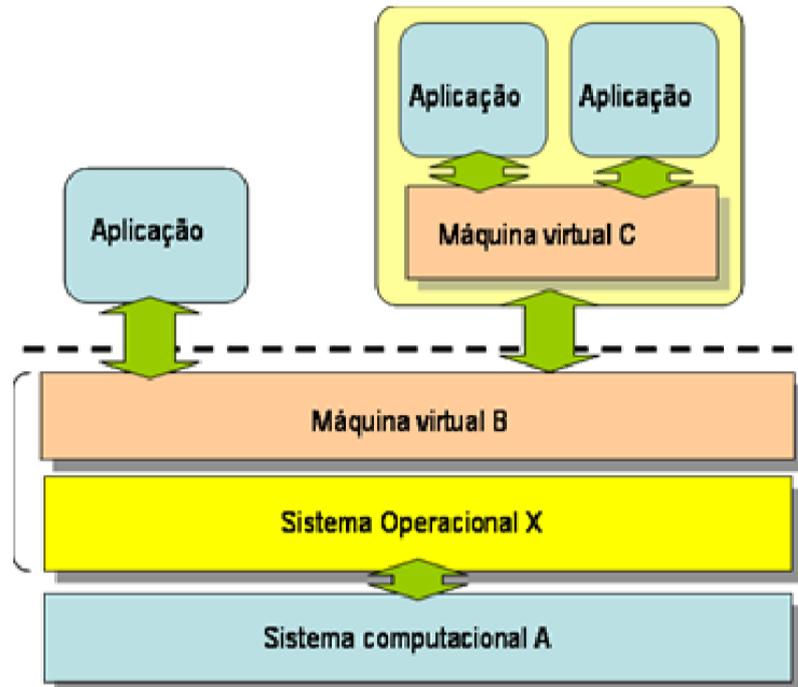
<sup>2</sup> Ambiente de trabalho gráfico desenvolvido e mantido através do projeto de software livre GNOME.

De acordo com Carissimi (2008), apesar da introdução de linguagens de programação como o Java trazerem consigo cenários do tipo, o emprego de máquinas virtuais (VMs - *Virtual Machines*) começou na década de 1970, com a virtualização de *mainframes* da IBM. Na época, era comum que cada *mainframe* contasse com seu sistema operacional específico e proprietário, o que levava a incompatibilidade de aplicações caso o hardware ou sistema fosse trocado. Tendo este problema como motivação, surge, então, o conceito de oferecer um ambiente virtual para a aplicação, simulando toda a estrutura de hardware e sistema operacional adequada para o seu correto funcionamento. Isso possibilitou a execução e migração de aplicações entre plataformas, contanto que existisse uma versão de máquina virtual para a plataforma almejada, eliminando grande parte dos problemas de incompatibilidade (CARISSIMI, 2008).

Veras (2011) explica que a virtualização desvincula o sistema operacional e aplicações do hardware, propiciando, assim, maior agilidade na instalação, configuração, gerenciamento e migração de ambientes em servidores. A flexibilidade que um ambiente desse tipo possibilita acaba tornando-o muito atrativo para diversos cenários. Com o uso da virtualização, por exemplo, ambientes de testes específicos ou que executem aplicações críticas podem ser facilmente criados, sendo mantidos isolados de outros serviços de um cluster ou servidor, possibilitando maior segurança e facilidade de gerenciamento.

Uma máquina virtual pode ser definida como um nível de software, que oferece um ambiente de hardware simulado muito parecido com uma máquina física, podendo cada máquina contar com sistema operacional, bibliotecas e aplicações próprias. A virtualização tem como objetivo simular o comportamento de uma interface ou recurso, conforme pode ser visto na Figura 16. Nela, um Sistema Operacional X, instalado sob o Sistema Computacional A, serve de base para uma Máquina Virtual B, que, por sua vez, simula um ambiente que suporta uma aplicação comum e outra aplicação que simula a Máquina Virtual C (CARISSIMI, 2008).

Figura 16 – Princípio básico de máquinas virtuais



Fonte: Carissimi (2008, p. 177).

### 2.6.1 Modos de implementação

O processo ou sistema que é executado dentro da máquina virtual é chamado de *guest* (convidado), enquanto a plataforma que a suporta é chamada de *host* (hospedeiro) (SMITH; NAIR, 2005).

De acordo com Smith e Nair (2005), a implementação de máquinas virtuais pode ser efetuada de duas maneiras distintas, conforme detalhado a seguir:

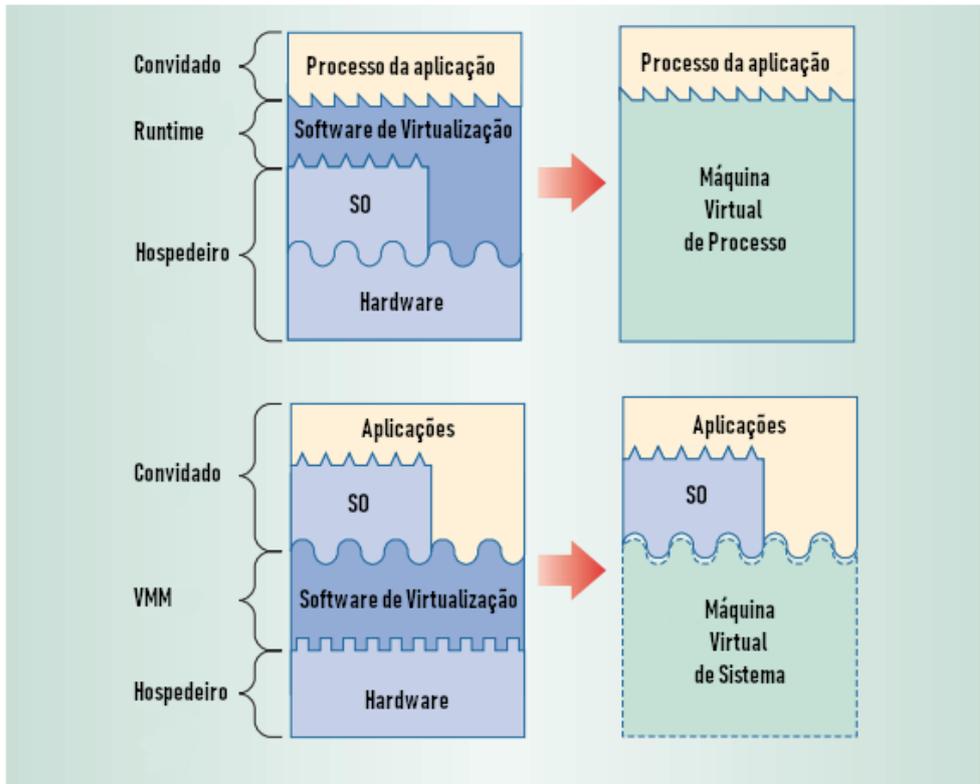
- a) Máquina Virtual de Processo:** neste tipo de implementação, o software de virtualização, chamado de *runtime*, está no nível de API ou *Application Binary Interface* (ABI), rodando sob uma combinação de sistema operacional e hardware do hospedeiro. O software irá emular instruções a nível de usuário, bem como outras chamadas do sistema operacional e bibliotecas. Normalmente, este tipo de máquina existe somente para suportar um processo, ou seja, só é criada para a execução de um programa e, logo depois, é eliminada. Carissimi (2008) cita como exemplo a máquina virtual Java (JVM),

bem como a execução do sistema operacional Linux em uma máquina hospedeira com Windows, através do software de virtualização *VMware Player*.

**b) Máquina Virtual de Sistema:** implementações deste tipo consistem em um software de virtualização, chamado de *Virtual Machine Manager* (VMM) ou *hypervisor*, rodando logo acima da camada de hardware do hospedeiro e antes do software convidado. Deve prover um ambiente capaz de suportar diversos sistemas operacionais convidados completos, disponibilizando recursos de hardware virtuais, como rede, processador, memória e interfaces de entrada/saída. Esse software deve estar disponível assim que o hospedeiro for ligado, podendo emular diferentes conjuntos de instruções de máquina de acordo com as necessidades dos softwares convidados, ou disponibilizar recursos de hardware virtualizados. Podemos citar como exemplos de *hypervisor* o Xen (KIVITY et al., 2007), o Microsoft Hyper-V (ALI, 2014) e o KVM (*Kernel Virtual Machine*) (KUDRYAVTSEV; KOSHELEV; AVETISYAN, 2012), o qual será apresentado em 2.6.2.

É possível observar uma representação simplificada de ambos os tipos de implementação na Figura 17.

Figura 17 - Representação dos dois modos de implementação de máquinas virtuais

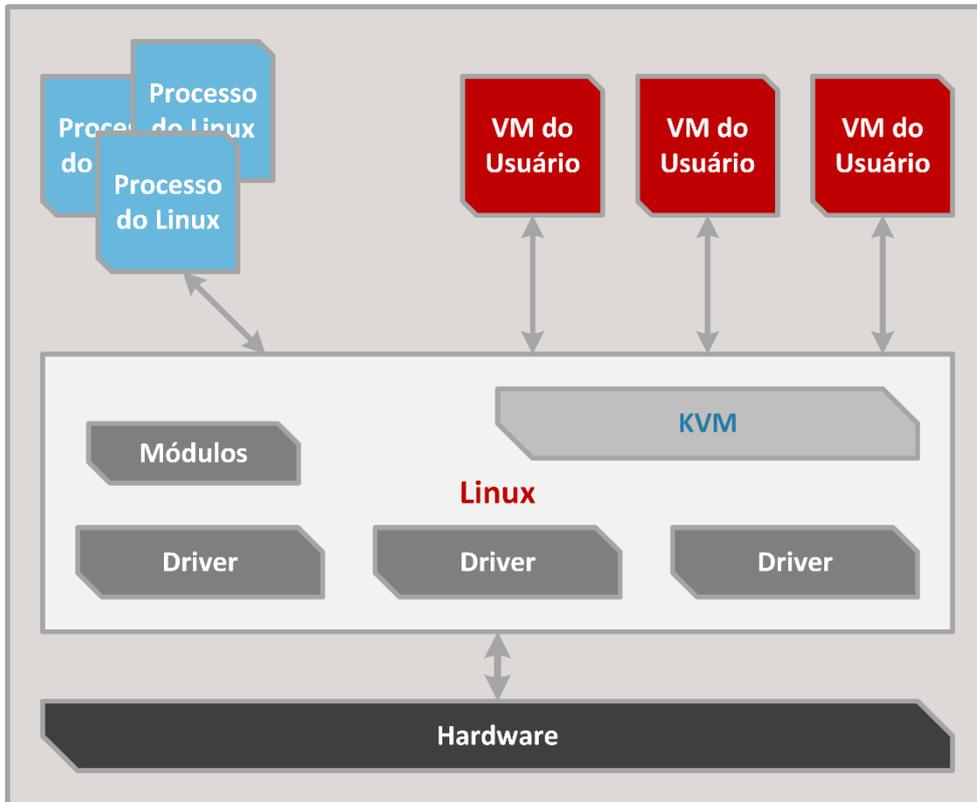


Fonte: Adaptada pelo autor, com base em Smith e Nair (2005).

## 2.6.2 KVM

A *Kernel-based Virtual Machine* (KVM, ou Máquina Virtual baseada em Núcleo) é um subsistema presente em várias distribuições do Linux, o qual tem o intuito de transformar o sistema em um *hypervisor* capaz de sustentar e gerenciar inúmeras máquinas virtuais, cada uma com seu *kernel* e sistema operacional independente. As máquinas virtuais aparecem como processos comuns do Linux, integrando-se perfeitamente ao sistema. A Figura 18 demonstra a arquitetura da KVM de forma simplificada (ALI, 2014; KIVITY et al., 2007).

Figura 18 – Arquitetura da KVM



Fonte: Adaptada pelo autor, com base em Herrmann et al. (2018).

### 2.6.2.1 Migração em tempo real

A migração em tempo real é a capacidade de poder mover a carga de trabalho de uma máquina virtual em execução entre um hospedeiro e outro, sem que haja interrupção ao usuário no uso desta. Essa característica é chave em sistemas de alta disponibilidade com balanceamento de carga, em que o *hypervisor* pode transferir a carga de um hospedeiro sobrecarregado para outro que esteja com mais recursos disponíveis (KIVITY et al., 2012).

Essa migração consiste na cópia da memória da máquina virtual convidada para o hospedeiro de destino enquanto está em execução. A KVM monitora qualquer mudança nas páginas de memória já transferidas através de uma solução chamada *dirty page log* (ou Log de Página Suja) e, caso ocorra alguma, efetua a cópia dessas alterações antes de finalizar a migração. A KVM também calcula o tempo estimado

dessa cópia e, caso necessário, pode suspender a máquina virtual por alguns milissegundos a fim de migrá-la completamente (KIVITY et al., 2012; HERRMANN et al., 2018);

A KVM também suporta a migração comum, ou *off-line*, em que uma máquina virtual é suspensa e, só então, sua memória é copiada e migrada para um novo hospedeiro. Herrmann et al. (2018) recomendam que este tipo de migração seja utilizado caso a máquina virtual modifique as páginas de memória mais rapidamente do que a KVM consiga migrá-las, visto que a migração em tempo real jamais seria completa nesse caso.

### **2.6.2.2 KVM no Red Hat Enterprise Linux**

Por se tratar de uma solução nativa, a KVM está totalmente integrada ao *kernel* do RHEL. Nesse sistema, ela é gerenciada pela API *libvirt*, um extenso conjunto de ferramentas suportado por diversas plataformas de virtualização, sendo a KVM uma delas. Essa API foca no gerenciamento de cada hospedeiro, provendo ferramentas para enumerar, monitorar e usar os recursos disponíveis neste, como CPU, memória e armazenamento (ALI, 2014; HERRMANN et al., 2018).

A KVM no RHEL também é integrada ao *Quick Emulator (QEMU) Guest Agent*, um agente que é executado na máquina convidada e que possibilita o recebimento de comandos do hospedeiro. O hospedeiro, nesse caso, pode, por exemplo, ativar e desativar CPUs virtuais enquanto a máquina virtual está em execução (HERRMANN et al., 2018).

## **2.7 Eficiência energética**

Nesta seção, são abordadas algumas iniciativas e métricas, as quais têm o intuito de aprimorar a eficiência energética em equipamentos de TI, seja somente em um cluster ou em um data center inteiro.

### 2.7.1 Green IT

O constante aumento no uso de equipamentos de TI no mundo todo associado às suas formas de produção e descarte, bem como a seu consumo elétrico vêm levantando preocupações quanto ao impacto destes dispositivos para o meio ambiente. O aumento do uso de eletricidade, emprego de materiais nocivos na produção, bem como o não reaproveitamento e/ou reciclagem desses equipamentos são alguns dos pontos que contribuem para o aumento dos gases do efeito estufa (MURUGESAN, 2008; RUTH, 2009).

A fim de diminuir este impacto, iniciativas como o Green IT (TI Verde) vêm ganhando força entre as fabricantes, organizações e usuários. Segundo Murugesan (2008), podemos resumir o Green IT em quatro objetivos, os quais são explanados de forma simplificada abaixo:

- **Uso verde:** deve-se reduzir o consumo de energia de computadores e outros equipamentos de TI, buscando usá-los de uma maneira ecologicamente correta e sadia;
- **Descarte verde:** deve-se tentar, ao máximo, reutilizar computadores e sistemas antigos e, caso não seja possível, providenciar a correta reciclagem e descarte de equipamentos não desejados;
- **Design verde:** deve-se projetar componentes, servidores, data centers e climatização que sejam energeticamente eficientes e que sejam ecologicamente corretos;
- **Fabricação verde:** deve-se fabricar componentes eletrônicos, computadores e outros equipamentos associados com o mínimo ou nenhum impacto ao meio ambiente, evitando, ao máximo, o uso de metais pesados e outros materiais nocivos;

Com esses objetivos em mente e como forma de estimular os fabricantes e organizações a adotarem essas medidas, surgiram padrões como o *Energy Star*, *Electronic Product Environmental Assessment Tool* (EPEAT) e o *80 PLUS*, além de

iniciativas como a *Green Grid*, que são explanadas nas subseções a seguir (MURUGESAN, 2008; KHAN et al., 2009).

### **2.7.1.1 Energy Star**

O *Energy Star* é um padrão norte-americano criado em 1992 pela *Environmental Protection Agency* (EPA), com o intuito de certificar equipamentos energeticamente eficientes. O padrão regula o consumo energético de fontes internas e externas de diversos equipamentos, durante todos os seus estados de atividade, como quando estão ativos, sem atividade ou em modo de espera (MURUGESAN, 2008; CALWELL; OSTENDORP, 2005).

O padrão *Energy Star* tornou-se uma certificação importante e reconhecida nos Estados Unidos e em muitos outros países, principalmente para equipamentos de TI, dentre computadores, servidores e monitores. A adoção ao programa de certificação teve muito sucesso e levou a significativas melhorias na eficiência energética dos equipamentos por ele abrangidos (KAHN et al., 2009; RUTH, 2009).

### **2.7.1.2 EPEAT**

O EPEAT é um padrão criado pelo *Green Electronics Council*, nos Estados Unidos, e tem por objetivo avaliar 55 critérios diferentes em produtos eletrônicos como computadores, monitores e servidores, a fim de avaliar seu impacto ambiental. Dentre esses critérios, são avaliados os materiais usados na produção, ciclo de vida do produto e conservação de energia elétrica (KATZ; RIFER; WILSON, 2005; MURUGESAN, 2008).

Os equipamentos avaliados pelo EPEAT recebem selos de bronze, prata ou ouro, de acordo com o atendimento dos critérios. Para um equipamento receber o selo de ouro, por exemplo, ele deve atender aos 22 dos critérios obrigatórios e, ao menos, 25 dos critérios opcionais. A adoção ao padrão por parte dos fabricantes é grande, tendo companhias como a Dell e HP na lista dos adeptos (KATZ; RIFER; WILSON, 2005; MURUGESAN, 2008).

### 2.7.1.3 80 Plus

Usualmente, uma fonte de energia em um computador converte a energia elétrica de corrente alternada, presente na rede elétrica, para corrente contínua, a qual é utilizada pelos equipamentos eletrônicos. Durante essa conversão, parte da energia é perdida, transformando-se em calor, e outra parte é utilizada efetivamente pelo equipamento (CALWELL; OSTERNDORP, 2005).

O programa *80 Plus* foi criado com o objetivo promover o desenvolvimento e comercialização de fontes de energia que sejam mais eficientes nesse processo de conversão e, conseqüentemente, mais eficientes energeticamente. Para receber a certificação *80 Plus*, uma fonte deve apresentar uma eficiência de, ao menos, 80% com cargas de 20%, 50% e 100%, e um fator de potência de, ao menos, 90% em carga máxima. Isso significa que 80% da energia de entrada deve poder ser utilizada pelo equipamento e não dissipada, além de dever oferecer ao menos 90% da sua potência nominal em carga máxima (KHAN et al., 2009).

### 2.7.1.4 Green Grid

De acordo com Veras (2010), o *Green Grid* é uma associação mundial composta por várias empresas do ramo de TI, criada com o objetivo de definir e promover boas práticas quanto ao consumo de energia em data centers. Já segundo Marin (2013), o *Green Grid* almeja constantemente criar métricas de avaliação e otimização da eficiência energética do data center, propondo soluções de implementação e melhorias com retorno em curto e longo prazo, estimulando gerentes de TI a adotarem medidas mais sustentáveis em suas estruturas.

De acordo com Veras (2010), a fim de poder calcular a eficiência de um data center e efetuar comparações quanto ao seu consumo energético foram criadas duas métricas pelo *Green Grid*, conforme segue:

- **Power Usage Efficiency (PUE):** índice obtido através da energia total consumida pela instalação dividida pela energia consumida pelos equipamentos de TI. Um PUE de 1.0 indica aproveitamento total da energia, já

um PUE de 3 indica que o data center necessita três vezes mais energia para se manter, dentre refrigeração e outros aspectos de infraestrutura, do que demandado por seus equipamentos de TI. O cálculo é expresso na Fórmula 1.

$$PUE = \frac{\text{Energia Total da Instalação}}{\text{Energia dos equipamentos de TI}} \quad (1)$$

- **Data Center Infrastructure Efficiency (DCiE):** métrica utilizada para representar a eficiência energética do data center. É expressa em porcentagem derivada do inverso do PUE, conforme Fórmula 2. Resultados maiores indicam melhor eficiência.

$$DCiE = \frac{1}{PUE} \quad (2)$$

De acordo com Marin (2013) e Veras (2010), essas métricas contribuem para o desenvolvimento de data centers com configurações mais eficientes, adotando, por exemplo, os paradigmas de clusters de baixo consumo e virtualização, explanados anteriormente neste trabalho, reduzindo também gastos com refrigeração e manutenção.

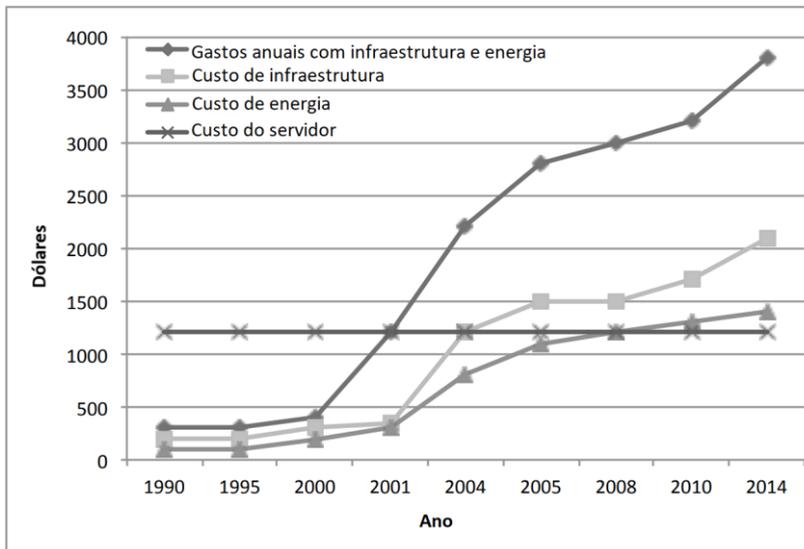
### 2.7.2 Green500

Durante muitos anos, o desenvolvimento dos supercomputadores buscava o desempenho a qualquer custo, ignorando o alto consumo de energia, além da necessidade de dissipação do crescente calor gerado por estes. A fim de comparação, desde 1992, o desempenho dos supercomputadores aumentou em torno de 10000 vezes, enquanto a medida de desempenho por watt aumentou somente em torno de 300 vezes (FENG; CAMERON, 2007).

Segundo Scogland, Subramaniam e Feng (2012), até 2001, adquirir um servidor de rack custava mais do que mantê-lo em operação durante um ano inteiro. Já em 2004, o valor gasto para mantê-lo em funcionamento durante o ano, incluindo custos de energia e infraestrutura, era o mesmo que seria gasto caso fosse comprado

um novo servidor. Por fim, em 2008, os gastos anuais com energia e infraestrutura ultrapassaram o valor de um servidor novo, conforme pode ser visto no Gráfico 1.

Gráfico 1 – Custos anuais em um data center com o passar dos anos



Fonte: Adaptado pelo autor, de Scogland, Subramaniam e Feng (2012).

Continuar com essa mentalidade no desenvolvimento de supercomputadores estava se mostrando cada vez mais inviável; portanto, com o objetivo de dar destaque ao problema presente, Wu-Chun Feng idealizou, em 2007, uma nova métrica para classificação de supercomputadores: flops/Watt. Essa métrica mede o número de operações de ponto flutuante que um supercomputador executa por Watt e estimulou a criação da lista Green500, em que os supercomputadores são classificados de acordo com essa métrica (FENG; CAMERON, 2007).

Diferentemente da lista TOP500, que só mensura o desempenho puro dos supercomputadores com base nos resultados do benchmark *High Performance Linpack* (HPL) e desconsidera seu consumo energético, a Green500 combina ambas as métricas (SCOGLAND; SUBRAMANIAM; FENG, 2012).

Pode-se observar, na Tabela 2, os dez supercomputadores mais eficientes do mundo segundo a lista Green500 de junho de 2018, bem como sua classificação na lista TOP500, seu desempenho no HPL em TFlops/s (mostrado na coluna 'Rmax'), seu consumo elétrico total expresso em kW e sua eficiência energética expressa em gigaflops/Watt (GFlops/W).

Tabela 2 – Os dez supercomputadores mais eficientes do mundo, segundo a lista Green500 de junho de 2018

Posição na Green500	Nome	Rmax (TFlop/s)	Energia Total (kW)	Eficiência elétrica (GFlops/W)	Posição na TOP500
1	Shoubu system B	857,6	47	18,404	359
2	Suiren2	798,0	47	16,835	419
3	Sakura	824,7	50	16,657	385
4	DGX SaturnV Volta	1070,0	97	15,113	227
5	Summit	122300,0	8806	13,889	1
6	TSUBAME3.0	8125,0	792	13,704	19
7	AIST AI Cloud	961,0	76	12,681	287
8	AI Bridging Cloud Infrastructure (ABCI)	19880,0	1649	12,054	5
9	MareNostrum P9 CTE	1018,0	86	11,865	255
10	RAIDEN GPU subsystem	1213,0	107	11,363	171

Fonte: Elaborada pelo autor, com base em TOP500 (2018).

É importante destacar que o supercomputador mais rápido do mundo segundo a TOP500, o Summit da IBM, posiciona-se em quinto lugar na lista Green500, o que demonstra o crescente interesse em desenvolver sistemas cada vez mais eficientes energeticamente. Ele utiliza processadores IBM POWER9 22C de 3,07 Giga-hertz (GHz) cada, chegando a uma contagem de 2.282.544 núcleos, aceleradores NVIDIA Volta GV100 e sistema Red Hat Enterprise Linux 7.4 (TOP500, 2018).

## **3 PROCEDIMENTOS METODOLÓGICOS**

Neste capítulo, são apresentados os procedimentos metodológicos empregados no desenvolvimento deste trabalho, a fim de possibilitar o atingimento dos objetivos propostos.

### **3.1 Quanto à natureza**

Quanto à natureza, este estudo se classifica como uma pesquisa aplicada. De acordo com Gil (2008, p. 27), este tipo de pesquisa tem como principal aspecto “o interesse na aplicação, utilização e consequências práticas dos conhecimentos”. Este tipo de pesquisa relaciona-se com a pesquisa pura, visto que faz uso de suas descobertas e se favorece com o seu progresso.

Para Mascarenhas (2012), este tipo de pesquisa parte da identificação de um problema e segue com a busca de soluções com o propósito de superá-lo. São fortemente relacionadas a aplicações práticas, baseando-se em percepções teóricas.

### **3.2 Quanto à abordagem**

O modo de abordagem empregado no trabalho foi o quali-quantitativo, ou seja, tanto a abordagem qualitativa quanto a quantitativa foram empregadas.

Para Prodanov e Freitas (2013), a pesquisa qualitativa tem como preceito que existe um vínculo entre o objetivo e o subjetivo de indivíduo, que não pode ser quantificado em números. Esse tipo de pesquisa não se utiliza de métodos estatísticos e foca na observação e entendimento de fenômenos, sendo os dados obtidos na presente pesquisa de caráter descritivo e analisados de forma indutiva pelo pesquisador.

Segundo Dalfovo, Lana e Silveira (2008), a pesquisa quantitativa é caracterizada por priorizar a quantificação, seja na recolha de dados bem como na sua análise, utilizando-se de métodos estatísticos de diferentes níveis de complexidade. Este tipo de pesquisa busca a exatidão, gerando resultados com poucas distorções.

### **3.3 Quanto aos objetivos**

Quanto aos objetivos, a pesquisa foi exploratória e descritiva, com o intuito de verificar e aplicar possíveis otimizações na configuração do cluster computacional, visando à melhoria da eficiência energética deste.

De acordo com Gil (2008), a pesquisa com caráter exploratório tem o intuito de expandir o conhecimento do pesquisador sobre determinado problema, propiciando uma melhor construção de hipóteses sobre ele. Pesquisas deste tipo envolvem levantamento bibliográfico, entrevistas e análise de dados obtidos.

Segundo Cervo, Bervian e da Silva (2007), pesquisas exploratórias são comumente utilizadas na formulação de hipóteses significativas para futuras pesquisas, sendo vista por determinados autores como quase científica.

Para Gil (2008), pesquisas de caráter descritivo objetivam-se na descrição das características de um fenômeno específico ou o estabelecimento de relações entre variáveis. Utilizam-se de meios padronizados para a coleta e análise de dados, a fim de suportar a pesquisa.

### 3.4 Quanto aos procedimentos técnicos

Os procedimentos técnicos adotados no trabalho foram a pesquisa bibliográfica e documental, aliadas à pesquisa experimental.

Segundo Prodanov e Freitas (2013), a pesquisa bibliográfica é aquela elaborada tendo como base materiais que já foram publicados, como livros, jornais, artigos científicos, internet, entre outros, deixando o pesquisador em contato direto com tudo já escrito sobre o tema da pesquisa. Mascarenhas (2012) destaca que a pesquisa deste tipo se baseia em informações já analisadas e publicadas, com embasamento científico, constituindo-se uma fonte segura de informação para o referencial teórico.

De acordo com Gil (2008), a pesquisa documental apresenta muitas semelhanças em relação à pesquisa bibliográfica, diferenciando-se apenas quanto à natureza das fontes de informações. A pesquisa documental se baseia em materiais que não receberam tratamento analítico, como documentos oficiais, reportagens de jornal, contratos, manuais de operações, dentre outros. Materiais que tiveram algum tipo de análise, como relatórios de empresas e pesquisas, também podem ser considerados.

Ainda segundo Gil (2008), a pesquisa experimental visa determinar e analisar uma ou mais variáveis que influenciem um objeto de estudo definido, estabelecendo modos de avaliação e de análise dos efeitos que estas possam vir a gerar no objeto. De acordo com Cervo, Bervian e da Silva (2007), a fim de atingir os resultados, o pesquisador utiliza-se de aparelhos e instrumentos que estão à sua disposição, bem como de procedimentos adequados que evidenciem a influência das variáveis sob o objeto de estudo. Neste trabalho, as variáveis principais de estudo foram o consumo de energia do cluster e a dissipação térmica gerada pelos componentes de hardware deste, evidenciando os fatores que pudessem vir a influenciá-las.

### 3.5 Coleta de dados

De acordo com Marconi e Lakatos (2017), a etapa de coleta de dados consiste na aplicação de técnicas específicas para recolha dos dados pressupostos. Para esse fim, Gil (2017) expõe que podem ser utilizados equipamentos mecânicos, elétricos ou eletrônicos.

No escopo do trabalho, a coleta de dados foi focada no consumo elétrico do cluster computacional, operando em momentos e cargas distintas. Foram efetuadas coletas em momentos de repouso, carga de uso média e carga de uso máxima, a fim de obter dados mais precisos da real eficiência energética do cluster sob várias situações.

Para a coleta dos dados do consumo energético do arranjo foram utilizadas ferramentas de gerenciamento do ambiente, como o *Integrated Dell Remote Access Controller* (iDRAC), as quais agregam dados de diversos sensores do hardware do cluster.

### 3.6 Análise dos dados

A etapa de análise dos dados se deu pelo tratamento e interpretação dos resultados obtidos, baseados nos métodos de coleta empregados (MARCONI; LAKATOS, 2017).

Os dados obtidos através das ferramentas de medição foram verificados e limpos, sendo após isso apresentados em tabelas e gráficos. Gil (2017) destaca que também podem ser empregados cálculos estatísticos como porcentagens, médias, correlações, dentre outros, segundo o objetivo da pesquisa.

A análise estatística destes dados possibilitou evidenciar a eficiência do cluster em relação ao seu consumo de energia elétrica e à sua capacidade de prover um ambiente de alta disponibilidade de forma eficaz.

### 3.7 Trabalhos relacionados

Vista a relevância de tópicos como eficiência energética em clusters e virtualização, é possível verificar um grande número de trabalhos publicados sobre o tema, com diferentes métodos de abordagem. A seguir são detalhados dois trabalhos relevantes sobre estes tópicos e relacionados ao tema desta pesquisa.

O trabalho desenvolvido por Gustavo Rostirolla, em 2014, teve por objetivo a análise de desempenho e consumo energético de um cluster computacional, formado por dez computadores de placa única com processadores ARM, os quais apresentam um reduzido consumo energético. Foram utilizadas placas BeagleBone Black para a estrutura do cluster desenvolvido pelo autor, a qual pode ser observada na Figura 19. O sistema operacional escolhido foi o Ubuntu Linux.

Figura 19 – Cluster desenvolvido com computadores de placa única



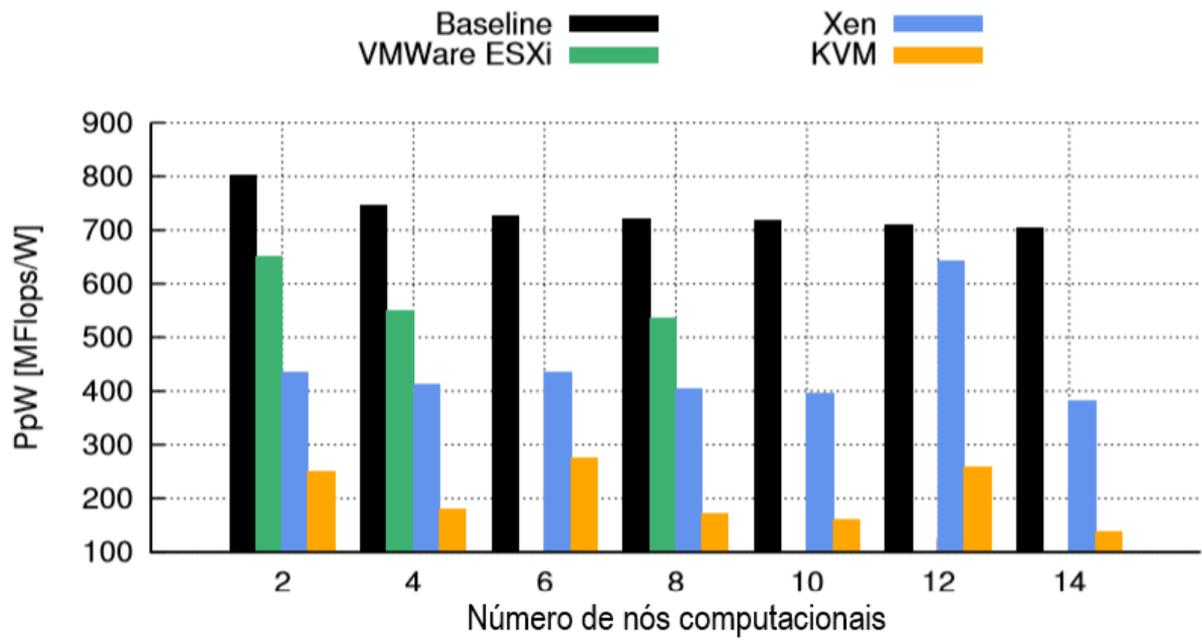
Fonte: Rostirolla (2014, p. 63).

Rostirolla (2014) efetuou experimentos utilizando softwares como o HPL e o Ondes3D, a fim de mensurar o desempenho e consumo energético do cluster sob diferentes cargas de trabalho, bem como verificar a viabilidade de um cluster do tipo. Ao final, o autor conseguiu atingir um desempenho de 27,33 Mflops/Watt e uma economia de energia de até 85,91% no momento em que nós ociosos do cluster eram desligados.

Outro estudo relacionado ao tema deste trabalho é o artigo publicado por um grupo de pesquisadores da Universidade de Luxemburgo. Varrete et al. (2013) efetuaram uma comparação entre três principais ferramentas de virtualização do mercado, a Xen, a KVM e a VMware ESXi, focando no seu desempenho e eficiência energética quando utilizadas em um HPC.

Os experimentos foram conduzidos no cluster *Taurus*, formado por 14 nós, sendo que cada nó possuía dois processadores Intel Xeon E5-2630 de 2,3 GHz, com 12 núcleos cada, 32 GB (Gigabyte) de memória RAM e sistema operacional Debian Linux. Foram criadas rotinas para criação das máquinas virtuais utilizando cada um dos *hypervisors*, para, em seguida, executar aplicações com alta demanda de CPU, memória e E/S, efetuando testes de stress em cada plataforma de virtualização. Paralelo a isso, a energia consumida em cada cenário também foi mensurada a cada segundo, através do uso de wattímetros e medidores precisos nas PDUs. Os dados de medição de cada *hypervisor* foram comparados a um *baseline*, que, no caso, se tratava de um cenário sem virtualização (VARRETE et al., 2013).

Ao final do estudo os autores concluíram que, apesar da virtualização oferecer vantagens em alguns cenários, o emprego de *hypervisors* gerou um impacto negativo na performance do HPC. Os cenários virtualizados também não proveram uma boa eficiência energética ao se deparar com aplicações de alta demanda como o HPL, não sendo uma boa opção para cenários de HPC. No Gráfico 2, pode-se verificar uma comparação da eficiência energética entre os três *hypervisors* e o *baseline* em execuções com diferentes quantidades de nós, utilizando como métrica a Performance por Watt (PpW), expressada em MFlops/W. É possível verificar que, em todos os cenários testados, a eficiência energética foi maior no que não contava com virtualização (VARRETE et al., 2013).

Gráfico 2 – Comparação da Performance por Watt entre cada *hypervisor* e o *baseline*

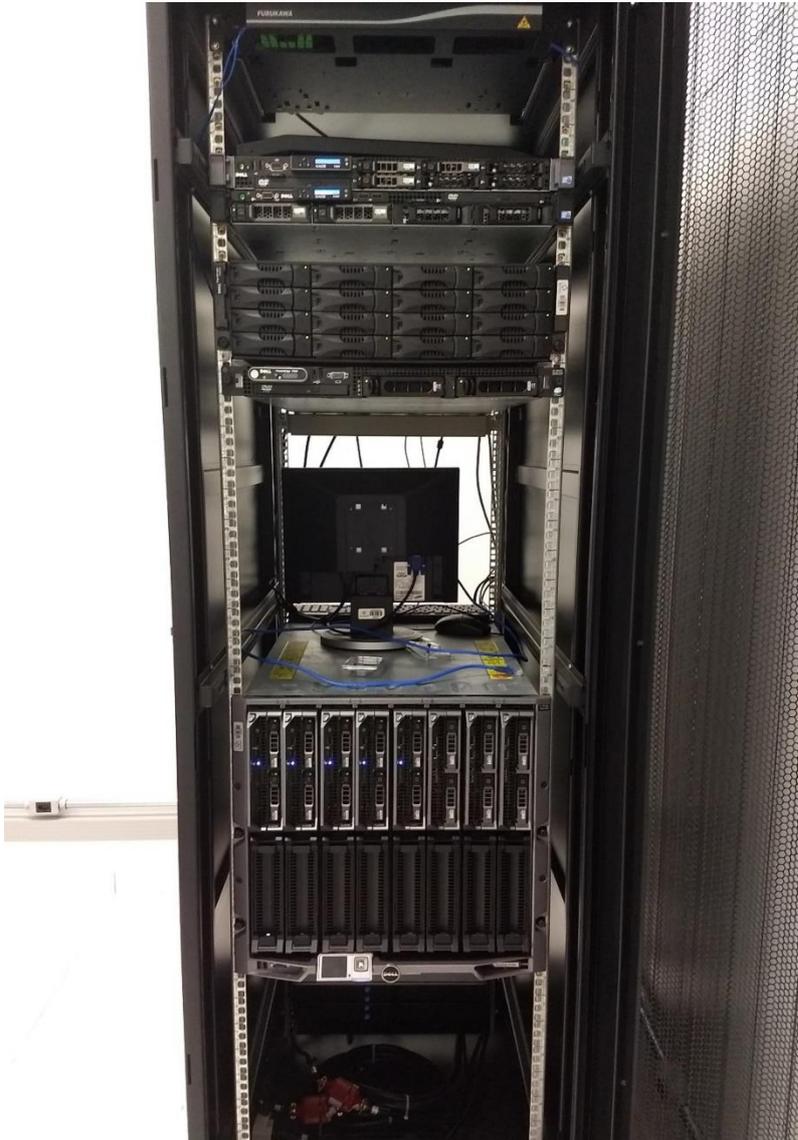
Fonte: Adaptado pelo autor, com base em Varrete et al. (2013).

## 4 CONFIGURAÇÃO DO AMBIENTE

Neste capítulo, são descritas as etapas efetuadas de configuração do cluster de alta disponibilidade e seus recursos, a fim de prover um cenário de testes para alcançar parte dos objetivos propostos neste trabalho. São abordadas etapas como a instalação do sistema operacional, a configuração do *storage*, rede, Pacemaker, entre outras. Também são apresentados maiores detalhes sobre o ambiente utilizado.

Como base para o cluster, foi utilizada a infraestrutura do data center educacional da Univates, o qual contém um rack composto por servidores do tipo blade da marca Dell, em conjunto com um sistema de armazenamento de dados de alto desempenho e duas unidades de distribuição de energia, PDU (*Power Distribution Unit*), adequadas. O rack pode ser visualizado na Figura 20.

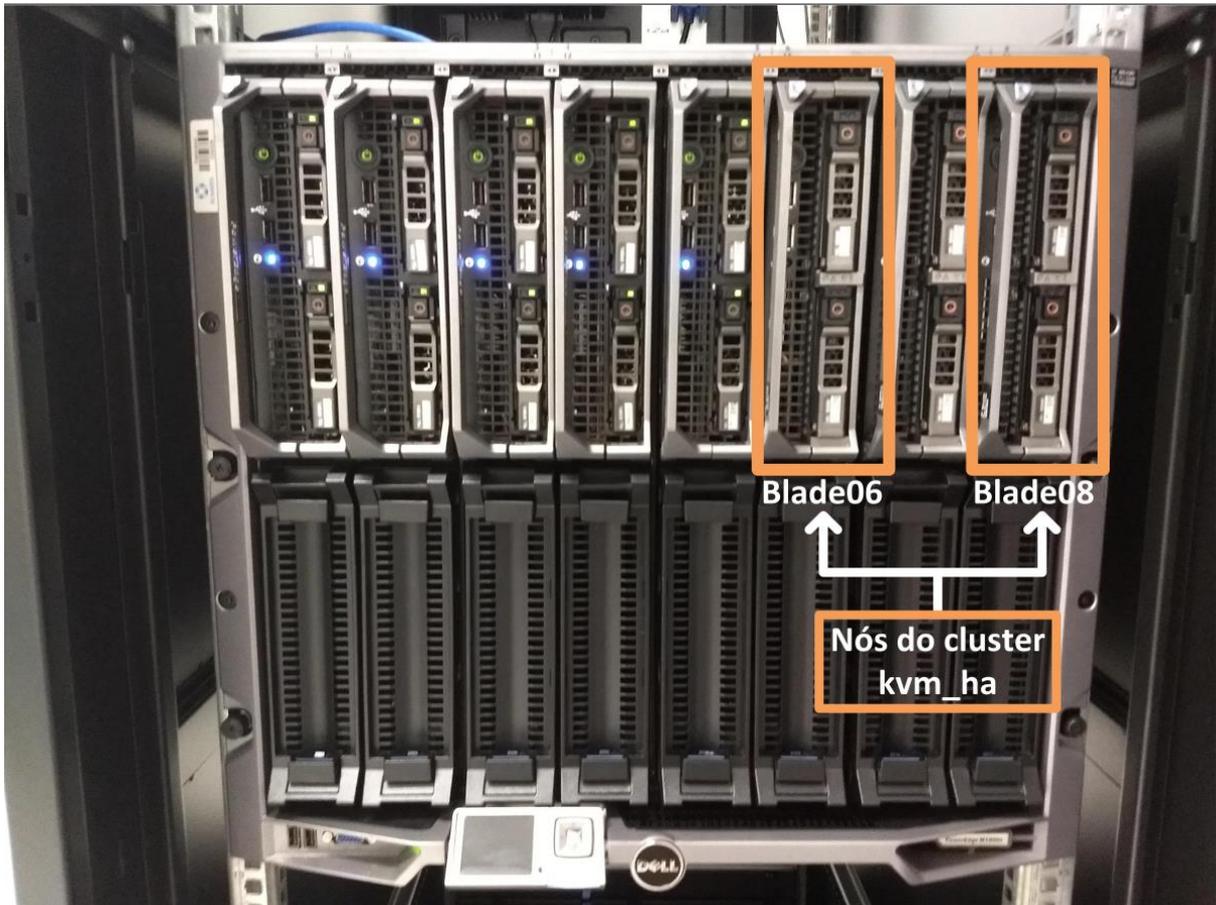
Figura 20 – Rack do data center educacional da Univates



Fonte: Do autor (2018).

O cluster é composto por dois nós, utilizando-se, para isso, dois servidores blade modelo PowerEdge M520, cujas configurações são: *blade06* – dois processadores Intel Xeon E5-2450 @ 2.10 GHz e 128 GB de memória RAM; e *blade08* – dois processadores Intel Xeon E5-2450 @ 2.50 GHz e 128 GB de memória RAM. Os blades ficam alocados em um chassi específico para eles; neste caso, é utilizado o modelo PowerEdge M1000e da Dell. Os servidores podem ser vistos na Figura 21.

Figura 21 – Chassi PowerEdge M1000e, destacando os dois nós do cluster configurado

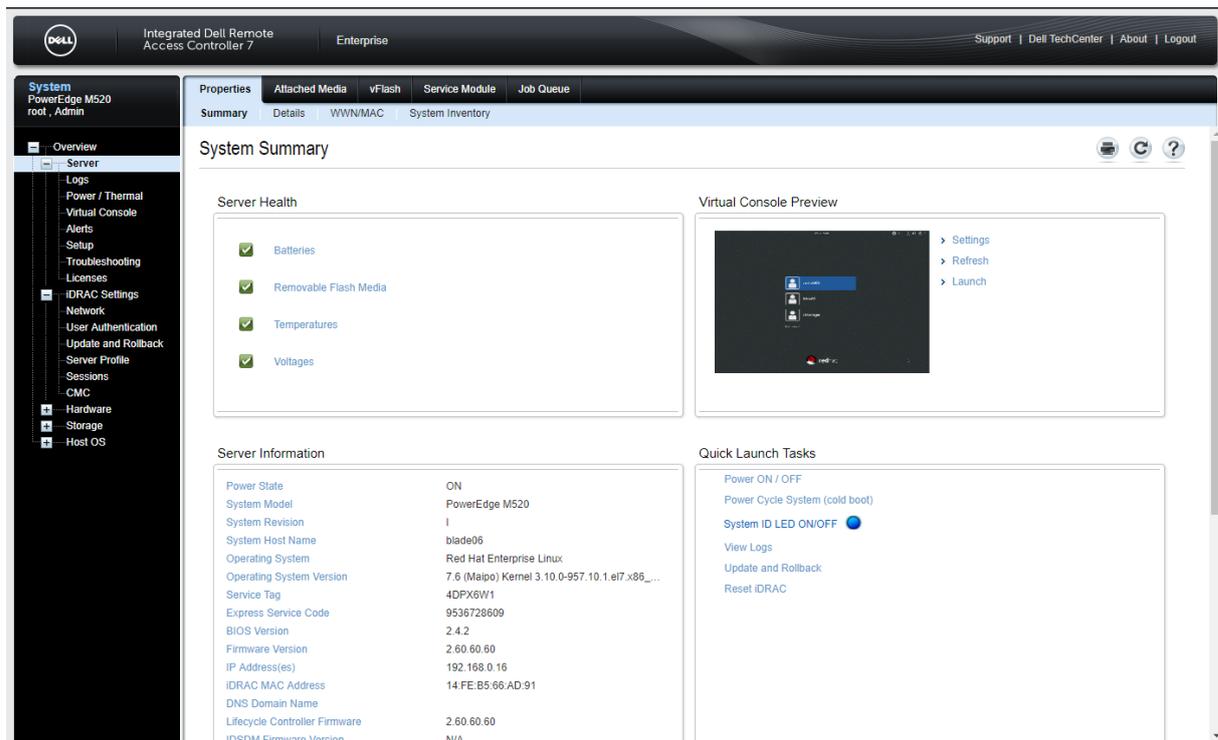


Fonte: Do autor (2018).

#### 4.1 Instalação do Red Hat Enterprise Linux 7.6

A fim de iniciar a instalação do sistema operacional em ambos os nós do cluster, foi necessário acessar o *Integrated Dell Remote Access Controller* (iDRAC), que é o sistema de gerenciamento *out-of-band* utilizado nos blades. Este fornece uma interface amigável, que permite acesso a opções de gerenciamento do hardware, a dados de consumo de energia e a um console remoto (DELL, 2018). A tela inicial do iDRAC pode ser visualizada na Figura 22 abaixo.

Figura 22 – Tela Inicial do iDRAC 7



Fonte: Do autor (2019).

Através do console remoto disponibilizado pelo iDRAC foi acessado, durante a inicialização do servidor, o *Lifecycle Controller*. Este fornece uma interface gráfica que permite gerenciar discos e atualizar *firmwares*, bem como configurar a rede de gerenciamento e a implantação de um sistema operacional (DELL, 2016). Através desta ferramenta, foi iniciada a instalação do Red Hat Enterprise Linux 7.6.

Cada blade conta com dois discos rígidos *Serial Attached SCSI<sup>3</sup>* (SAS) de 300 GB integrados, os quais foram configurados em *Redundant Array of Independent Disks* (RAID) 1, configuração esta que provê total redundância dos dados armazenados e possibilita *hot-swap*, que é a troca de um dos discos enquanto o sistema está ligado e operando. O sistema operacional foi instalado nesses discos locais.

Para instalação do RHEL 7.6, foi utilizada a interface gráfica padrão da Red Hat, a *Anaconda*, a qual possibilita a parametrização de discos locais, criação de

<sup>3</sup> Protocolo serial ponto a ponto de alto desempenho, utilizado comumente em servidores para conexão de discos rígidos e fitas de backup. Usa um conector serial similar ao SATA, sendo uma evolução do barramento SCSI paralelo criado nos anos 1980.

usuários, escolha dos recursos a serem instalados, entre outros. No cenário deste projeto, foi efetuada uma instalação completa com *Graphic User Interface* (GUI – Interface Gráfica do Usuário), abrangendo todos os pacotes necessários disponíveis na imagem de disco utilizada, a fim de facilitar o gerenciamento e configuração do sistema. Quanto aos usuários, além do padrão *root*, foram criados o usuário *arnhold06* para o *blade06*, o usuário *arnhold08* para o *blade08* e o usuário *clmanager*, este último a fim de facilitar o gerenciamento unificado do cluster.

#### 4.1.1 Configuração de rede e segurança

Após a instalação do sistema operacional, o próximo passo foi configurar as interfaces de rede, a fim de possibilitar a comunicação de cada blade entre si, com a *Storage Area Network*<sup>4</sup> (SAN) e com o mundo externo. Para isso, foi utilizada a interface gráfica do editor de conexões do *Network Manager*, pacote este que gerencia a rede no RHEL 7.6, acessível através do comando “`#nm-connection-editor`”.

Foram configuradas três interfaces de rede, detalhadas abaixo:

- Cluster LAN: interface Ethernet de 1 Gbit/s, com endereço IP 192.168.0.36 (*blade06*) e 192.168.0.38 (*blade08*). Esta rede é destinada à comunicação local entre os nós do cluster e com a rede de gerenciamento dos iDRACs e chassi dos blades;
- WAN: interface Ethernet de 1 Gbit/s, com endereço de IP público 177.44.248.29 (*blade06*) e 177.44.248.30 (*blade08*). Esta rede é destinada à comunicação de cada nó do cluster com a internet, possibilitando acesso externo e gerenciamento remoto;
- iSCSI VLAN: interface *enhanced Small Form-factor Pluggable* (SFP+) 10 Gigabit Ethernet (10-GbE), sob a qual foi configurada uma *Virtual LAN* (VLAN) com identidade 172 e atribuído: o IP 172.16.0.36 para o *blade06*; e o IP

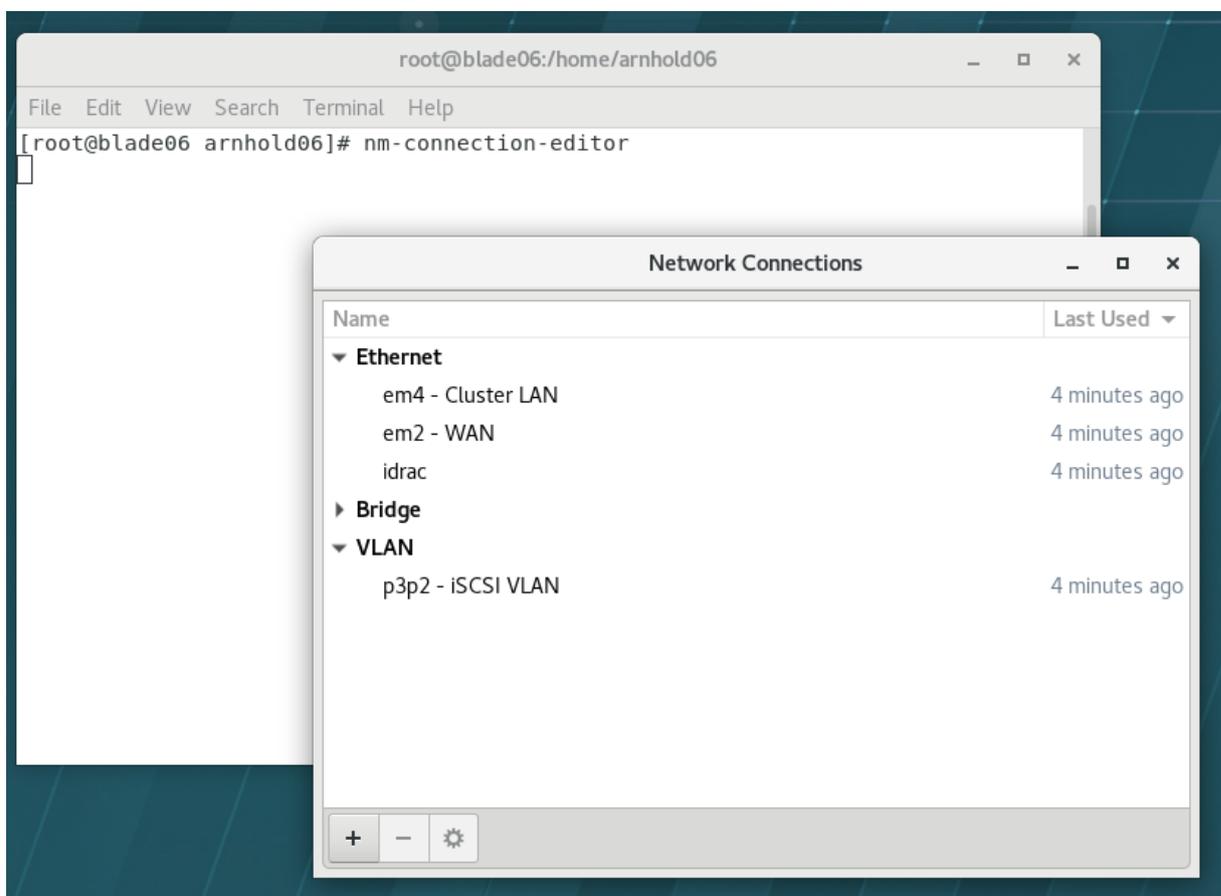
---

<sup>4</sup> Rede local de alto desempenho que provê acesso exclusivo a dispositivos de armazenamento pelos servidores.

172.16.0.38 para o *blade08*. Esta rede utiliza *fibre channel*<sup>5</sup> a fim de atingir maiores velocidades, sendo destinada à comunicação entre cada blade e o *array* de discos do *storage* SAN, que será detalhado em 4.1.2.

Pode-se verificar a tela inicial da interface gráfica do editor de conexões do *Network Manager* na Figura 23 abaixo, juntamente com um console mostrando o comando necessário para executá-la.

Figura 23 – Interface Gráfica do Editor de Conexões do Network Manager



Fonte: Do autor (2019).

A interface de rede intitulada “idrac” foi criada automaticamente pelo agente do iDRAC, instalado no sistema operacional. Este agente permite a comunicação direta entre o SO e o iDRAC, compartilhando informações como status e IP de cada interface

---

<sup>5</sup> Tecnologia de comunicação de alta velocidade que é utilizada em armazenamento de dados em rede, a partir de links de fibra óptica.

de rede. A rede tem o IP 169.254.0.2 atribuída a ela, em ambas os blades, e utiliza um *Network Interface Controller* (NIC) USB interno, não acessível pelo usuário.

Após a configuração de rede, foi ativado o protocolo *Secure Shell* (SSH) na porta 22, a fim de possibilitar o gerenciamento remoto e a comunicação entre os nós do cluster via *Corosync Cluster Engine*<sup>6</sup>. Ao configurar endereços de IP públicos e habilitar o SSH em ambos os blades, estes ficaram suscetíveis a ataques maliciosos, sendo necessária a configuração de um meio de proteção que impedisse intrusões.

Foram verificadas várias tentativas de conexões não autorizadas via SSH a partir do método de ataque, conhecido como força bruta, o qual testa diversas combinações de usuários e senhas. Caso as credenciais de acesso sejam fracas e nenhuma proteção esteja configurada, um invasor pode facilmente tomar controle do sistema, roubando dados ou tornando-o inutilizável (JAVED; PAXSON, 2013).

Para prevenir tais ataques, foi instalado o *Fail2Ban*, um *Intrusion Prevention System* (IPS) gratuito e de fácil utilização. Foi configurado um filtro que bloqueia por vinte e quatro horas qualquer IP que tente se conectar por mais de três vezes via SSH com credenciais inválidas. Na Figura 24, é mostrado o status do filtro “sshd” do Fail2Ban no *blade06*, apresentando todos os endereços de IP bloqueados naquele momento.

---

<sup>6</sup> Projeto derivado do OpenAIS, liberado sob a licença BSD, com objetivo desenvolver, liberar e dar suporte a um cluster de código aberto.

Figura 24 – Filtro “sshd” do Fail2Ban, apresentando os IPs bloqueados no momento

```

root@blade06:~
[root@blade06 ~]# fail2ban-client status sshd
Status for the jail: sshd
|- Filter
| |- Currently failed: 18
| |- Total failed:      5006
| `-- Journal matches:  _SYSTEMD_UNIT=sshd.service + _COMM=sshd
`-- Actions
   |- Currently banned: 218
   |- Total banned:     1617
   `-- Banned IP list:  182.61.34.79 117.50.46.176 106.13.121.175 183.107.101.23
8 196.223.156.1 172.245.135.162 37.187.193.19 103.108.144.134 104.236.81.204 123
.207.124.214 123.207.153.187 51.79.129.245 182.23.42.196 206.189.86.17 45.55.41.
232 147.135.27.166 114.118.91.64 132.255.29.228 167.99.161.15 115.231.174.45 188
.153.91.52 105.235.116.254 51.38.179.143 142.93.198.86 196.203.31.154 173.174.20
3.206 207.154.232.160 74.63.232.2 5.196.110.170 185.244.25.205 46.101.27.6 188.1
31.170.119 182.218.64.111 184.73.200.178 159.89.139.228 96.76.166.105 178.128.10
7.61 144.217.161.78 64.76.6.126 196.11.231.220 142.44.247.87 68.183.31.138 144.2
17.237.117 51.15.162.223 121.238.107.61 54.39.145.123 180.250.162.9 122.55.19.11
5 94.183.251.239 49.247.203.205 188.15.100.200 178.62.224.96 128.199.182.235 134
.209.99.242 206.189.145.152 62.197.120.198 51.38.36.84 176.31.24.51 175.6.66.48
185.216.25.94 180.167.198.186 128.199.133.249 187.32.178.33 163.125.135.143 159.
65.245.203 192.241.249.19 202.117.7.130 222.127.30.130 206.189.88.75 189.34.59.9
7 84.93.153.9 198.27.67.173 212.98.190.248 142.93.39.29 202.196.52.8 188.131.146
.22 104.238.81.58 159.89.205.213 91.121.226.117 193.153.174.183 202.22.142.111 1
28.199.100.253 198.245.49.37 51.75.202.218 54.36.165.226 54.37.14.3 65.26.240.14
138.68.146.186 165.22.246.5 159.65.236.58 89.184.67.90 144.217.4.14 104.37.79.1

```

Fonte: Do autor (2019).

#### 4.1.2 Mapeamento do armazenamento SAN

A fim de prover armazenamento em disco compartilhado entre os nós do cluster, foi utilizado um *storage* SAN da marca Dell, modelo EqualLogic PS6010. Ele é composto por um *array* de 16 discos rígidos SAS de 600 GB, os quais atingem 15000 rotações por minuto (RPM), sendo específicos para ambientes de alto desempenho. O *storage* está configurado em RAID 6, arranjo este que provê maior redundância de dados em caso de falha de um ou mais discos. A parte frontal do *storage* é mostrada na Figura 25 abaixo.

Figura 25 – SAN Dell EqualLogic PS6010 utilizada



Fonte: Do autor (2018).

Visto que o rack onde foi configurado o cluster é compartilhado com outros acadêmicos e professores da instituição, não foi utilizado todo o *storage* para o cluster, e sim criado um volume de 1 TB, acessível e compartilhado por ambos os nós, chamado “LAMINA-6-8”. A SAN é conectada ao *switch* presente no chassi do conjunto de blades através de uma rede 10-GbE, operando sob a VLAN 172, citada anteriormente.

A conexão entre o volume criado e os nós do cluster é realizada através do protocolo *Internet Small Computer System Interface*<sup>7</sup> (iSCSI). Nesse protocolo, o volume recebe um endereço chamado *target*, o qual é utilizado pelo sistema operacional para mapeá-lo através do IP do *array*.

O mapeamento pelo RHEL do volume iSCSI é obrigatório em ambos os nós do cluster, sendo necessária a criação um iSCSI *initiator*. Para isso, foram utilizados os comandos a seguir:

- # `yum install iscsi-initiator-utils -y`: instala o pacote com utilitários iSCSI no RHEL;
- # `iscsiadm -m discovery -t st -p 172.16.0.10`: retorna o endereço do target iSCSI, presente no IP informado;

<sup>7</sup> Protocolo de rede comum às conexões SAN, que torna mais fácil a transferência de informações através da implementação da tecnologia SCSI sobre a rede IP.

- `# iscsiadm -m node -T iqn.2001-05.com.equallogic:0-8a0906-2d1e9e70a-faf00000245b6e0-lamina-8 -l: efetua login no target informado, com o endereço descoberto no comando anterior;`
- `# grep "Attached SCSI" /var/log/messages: retorna o nome do disco iSCSI – ao qual, neste caso, foi atribuído o nome "/dev/sdb" – já mapeado pelo sistema.`

Após seguir esses comandos, o disco já está mapeado pelo sistema e pode ser formatado para uso compartilhado pelo cluster. O processo de criação do sistema de arquivos e integração do *storage* como recurso do cluster será detalhado em 4.2.1.2.

## 4.2 Criação e configuração do cluster

Após a configuração inicial do sistema, a próxima etapa foi criar o cluster em si. Para isso, inicialmente, foi necessário instalar os pacotes “*pcs*”, “*pacemaker*” e “*fence-agents-all*” através do comando “`# yum install pcs pacemaker fence-agents-all`”. Esses pacotes, essenciais para a criação e o funcionamento do cluster, são detalhados a seguir:

- *pcs*: é uma ferramenta de configuração para o Corosync e o Pacemaker, que permite fácil criação e gerenciamento do cluster, além de conter o *pcsd*, um *daemon* que opera como um servidor remoto e fornece uma interface de usuário na web. Essa interface possibilita um controle facilitado dos principais parâmetros do cluster;
- Pacemaker: atua como um *Cluster Resource Manager* (CRM), que oferece uma interface de gerenciamento do cluster, controlando seus recursos e tomando as decisões após a formação deste pelo Corosync;
- Corosync: instalado como dependência do *pcs*, o Corosync é responsável por formar e monitorar o estado dos nós do cluster, implementando, para isso, o *Totem Single Ring Ordering and Membership Protocol*. Este protocolo troca *tokens* entre os nós do cluster, a fim de determinar o estado deles: caso um nó deixe de

responder o *token*, ele é excluído do cluster e este é reformulado. Também é responsável pela troca de mensagens entre os nós e pela formação do quórum, usando tunelamento SSH para comunicação.

Após a instalação dos pacotes, foi necessária a liberação do serviço de alta disponibilidade no firewall do sistema de cada blade. Para isso, foram executados os seguintes comandos:

```
# firewall-cmd --permanent --add-service=high-availability
# firewall-cmd --add-service=high-availability
```

Liberadas todas as portas necessárias, foi definida uma senha padrão para o usuário “hacluster”, a conta administrativa do *pcs*, em ambos os nós. Na sequência, o serviço do *pcsd* foi iniciado e definido para iniciar automaticamente após o *boot* do sistema. Com esses passos concluídos, foi feita a autenticação do usuário “hacluster” no nó que irá executar o *pcs*, através do comando “# *pcs cluster auth blade06 blade08*”. Essa autenticação permite controle dos nós pelo *pcs*, sendo possível iniciar ou parar ou cluster inteiro, bem como outras funções, através de um único nó.

Por fim, foi executado o comando para criação do cluster – # *pcs cluster setup --start --kvm\_ha\ blade06 blade08* – especificando o nome do cluster “*kvm\_ha*” e o *hostname* dos nós participantes: *blade06* e *blade08*. Feito isso, o cluster foi criado e iniciado, mas ainda sem recursos, cuja criação e configuração são detalhadas nas subseções a seguir.

## 4.2.1 Recursos do cluster

Nas subseções a seguir, são descritos os recursos implementados no cluster, detalhando sua configuração e objetivos.

### 4.2.1.1 Isolamento do cluster – *fencing*

Os dois primeiros recursos configurados no cluster foram recursos de isolamento, também chamados de *fencing*. Eles têm por objetivo isolar um nó do

cluster caso este deixe de responder, a fim de evitar que o nó defeituoso continue escrevendo no disco, o que poderia causar corrupção dos dados.

O isolamento é efetuado utilizando o método *Shoot The Other Node In The Head* (STONITH - Atirar Na Cabeça Do Outro Nó). Nesse método, o nó defeituoso é reiniciado ou desligado à força, removendo-o do cluster. A fim de possibilitar esse comportamento, foram configurados dois recursos, um para cada nó, intitulados “ipmi-fence-blade06” e “ipmi-fence-blade08”.

Esses recursos utilizam a interface de gerenciamento *Intelligent Platform Management Interface* (IPMI) presente em cada blade, acessível via LAN através do IP do iDRAC, para reiniciar o nó caso este venha a parar de responder ao Corosync e for considerado defeituoso. Em cada iDRAC, foi configurado um usuário de nível administrador, com o nome *fencer*, a fim de possibilitar o acesso do recurso à interface. O comando utilizado para configuração do recurso “ipmi-fence-blade06” é explicitado abaixo:

```
# pcs stonith create ipmi-fence-blade06 fence_ipmilan
pcmk_host_list="blade06" ipaddr="192.168.0.16" login="fencer"
passwd="fencer123" lanplus=1 power_wait=4
```

#### 4.2.1.2 Sistema de arquivos compartilhado - GFS2

Os recursos que foram configurados na sequência têm por objetivo disponibilizar um sistema de arquivos para o volume compartilhado montado em ambos os nós, o qual permita gravação e leitura simultâneas pelos outros recursos do cluster, sem gerar inconsistências. A fim de prover tal cenário, foi utilizado o sistema de arquivos recomendado pela Red Hat para esse tipo de configuração, o *Global File System 2* (GFS2).

Com o propósito de configurar o GFS2, primeiramente, foi necessária a instalação do *add-on Resilient Storage*, disponível na imagem de disco do RHEL 7.6, o qual contém os pacotes “lvm2-cluster” e “gfs2-utils”. O pacote “lvm2-cluster” instala o *Clustered Logical Volume Manager* (CLVM), uma extensão ao *Logical Volume Manager* (LVM), a qual habilita o uso de volumes lógicos (*Logical Volumes*, ou LV) em

ambientes de cluster, provendo diversas ferramentas, como o *Cluster Logical Volume Manager Daemon* (CLVMD) e *Distributed Lock Manager* (DLM).

Após a instalação dos pacotes, foi necessário mudar a política de perda de quórum do cluster, a qual, por padrão, quando o quórum é perdido, faz com que o Pacemaker pare os recursos. Como o GFS2 necessita de quórum para se manter em execução, a política foi alterada para “freeze”, através do comando abaixo:

```
# pcs property set no-quorum-policy=freeze
```

Ao aplicar tal política o cluster irá congelar os recursos em caso de perda de quórum, evitando a falha do GFS2 e um consequente isolamento forçado do cluster toda vez que este falhasse. Também foi necessário ativar o DLM em ambos os nós, alterando o parâmetro *locking-type* constante no arquivo “/etc/lvm/lvm.conf” para 3, através do uso do comando a seguir:

```
# /sbin/lvmconf --enable-cluster
```

O DLM age gerando travas para arquivos ou pastas que estejam em edição por um determinado nó, evitando que o outro nó possa vir a tentar editá-lo também e causar corrupção, sendo pré-requisito para o correto funcionamento do CLVM e do sistema de arquivos GFS2. Foram criados dois recursos clones intitulados “*dlm*”, sendo que cada um irá executar em um dos nós, utilizando o comando abaixo:

```
# pcs resource create dlm ocf:pacemaker:controld op monitor interval=30s on-fail=fence clone interleave=true ordered=true
```

Logo após, foi efetuada a criação de mais dois recursos clones intitulados “*clvmd*”, a fim de possibilitar que o Pacemaker gerencie o CLVMD através do agente de recurso *ocf:heartbeat:clvm*. Este fica responsável por iniciar os *daemons clvmd* e *cmirror*, além de garantir que o parâmetro *locking\_type*, citado anteriormente, esteja definido corretamente. O comando para criação desse recurso pode ser visto abaixo:

```
# pcs resource create clvmd ocf:heartbeat:clvm op monitor interval=30s on-fail=fence clone interleave=true ordered=true
```

Visto que o recurso “*clvmd*” é dependente do “*dlm*”, foi necessário definir uma regra de dependência, a fim destes recursos iniciarem na ordem correta e nos nós

corretos. O Pacemaker chama tais regras de *constraints*, e os comandos utilizados para defini-las são apresentados a seguir:

```
# pcs constraint order start dlm-clone then clvmd-clone
# pcs constraint colocation add clvmd-clone with dlm-clone
```

Por fim, foi efetuada a criação do volume lógico clusterizado, o qual foi formatado para o sistema de arquivos GFS2. Os comandos utilizados são especificados a seguir:

- # pvcreate /dev/sdb: cria o volume físico (*Physical Volume*, ou PV) com o volume iSCSI montado;
- # vgcreate -Ay -cy clustervg /dev/sdb: cria o grupo de volumes (*Volume Group*, ou VG) “clustervg” com o PV criado;
- # lvcreate -L900G -n clusterlv clustervg: cria o volume lógico (LV) “clusterlv” com o VG criado;
- # mkfs.gfs2 -j2 -p lock\_dlm -t kvm\_ha:cluster\_gfs2 /dev/clustervg/clusterlv: formata o LV para o sistema de arquivos GFS2 e define o tipo de trava a ser utilizado;

Com a criação do volume lógico e sua formatação concluídas, são criados, por fim, dois recursos clones para gerenciar o sistema de arquivos, denominados “clusterfs”. Na criação desse recurso, foi especificado que o volume será montado em uma pasta intitulada “/shared” em cada nó do cluster. O comando utilizado para a criação desse recurso é explicitado abaixo:

```
# pcs resource create clusterfs Filesystem
device="/dev/clustervg/clusterlv" directory="/shared"
fstype="gfs2" "options=noatime" op monitor interval=10s on-
fail=fence clone interleave=true'
```

Dada a dependência do recurso “clvmd” pelo “clusterfs”, também foi necessária a configuração de uma regra para a correta inicialização do sistema de arquivos, a fim de garantir que o recurso “clusterfs” inicie após o “clvmd”. Após essas etapas, o sistema de arquivos foi montado e disponibilizado como um recurso do cluster, sendo somente acessível enquanto o cluster estiver em execução.

### 4.2.1.3 Endereço de IP virtual

Com o intuito de abstrair o cluster para o usuário final, a fim deste ser percebido como uma máquina só, foi efetuada a configuração de um endereço de IP virtual. Para disponibilizar este IP, foi criado o recurso “virtual\_ip”, o qual utiliza o agente de recurso *ocf:heartbeat:IPAddr2*, e designado o IP público 177.44.248.28.

O recurso foi configurado para se manter em execução no nó ativo, direcionando o acesso ao IP real deste. Na Figura 26, é mostrada a interface web do *pcs* sendo acessada através do endereço de IP virtual.

Figura 26 – Interface web do *pcs* acessada através do IP virtual

The screenshot displays the Pacemaker/Corosync Configuration web interface. The browser address bar shows the URL: `https://177.44.248.28:2224/managec/kvm_ha/main#/resources/virtual_ip`. The interface includes a navigation menu with tabs for NODES, RESOURCES, FENCE DEVICES, ACLS, CLUSTER PROPERTIES, and MANAGE CLUSTERS. The main content area is titled "RESOURCES" and shows a list of resources. The "virtual\_ip" resource is selected and its configuration is displayed in the "Edit Resource virtual\_ip" panel.

NAME	TYPE
clusterfs-clone	Clone
clvmd-clone	Clone
dlm-clone	Clone
virtual_ip	ocf:heartbeat:IPAddr2
VM_ws2016_01	ocf:heartbeat:VirtualDomain
VM_ws2016_02	ocf:heartbeat:VirtualDomain
VM_ws2016_03	ocf:heartbeat:VirtualDomain
VM_ws2016_04	ocf:heartbeat:VirtualDomain
VM_ws2016_05	ocf:heartbeat:VirtualDomain
VM_ws2016_06	ocf:heartbeat:VirtualDomain

**Edit Resource virtual\_ip**

virtual\_ip ✓ running

Enable  
  Disable  
  Cleanup  
  Refresh  
  Remove  
 Manage  
  Unmanage

Type: ocf:heartbeat:IPAddr2  
 Description: Manages virtual IPv4 and IPv6 addresses (Linux specific version)

Current Location: blade06

Clone:

Master/Slave:

Group: None

▼ Resource Location Preferences (1)

Node/Rule	Score	Remove
blade06	INFINITY	<input checked="" type="checkbox"/>
<input type="text"/>	<input type="text"/>	<input type="button" value="Add"/>

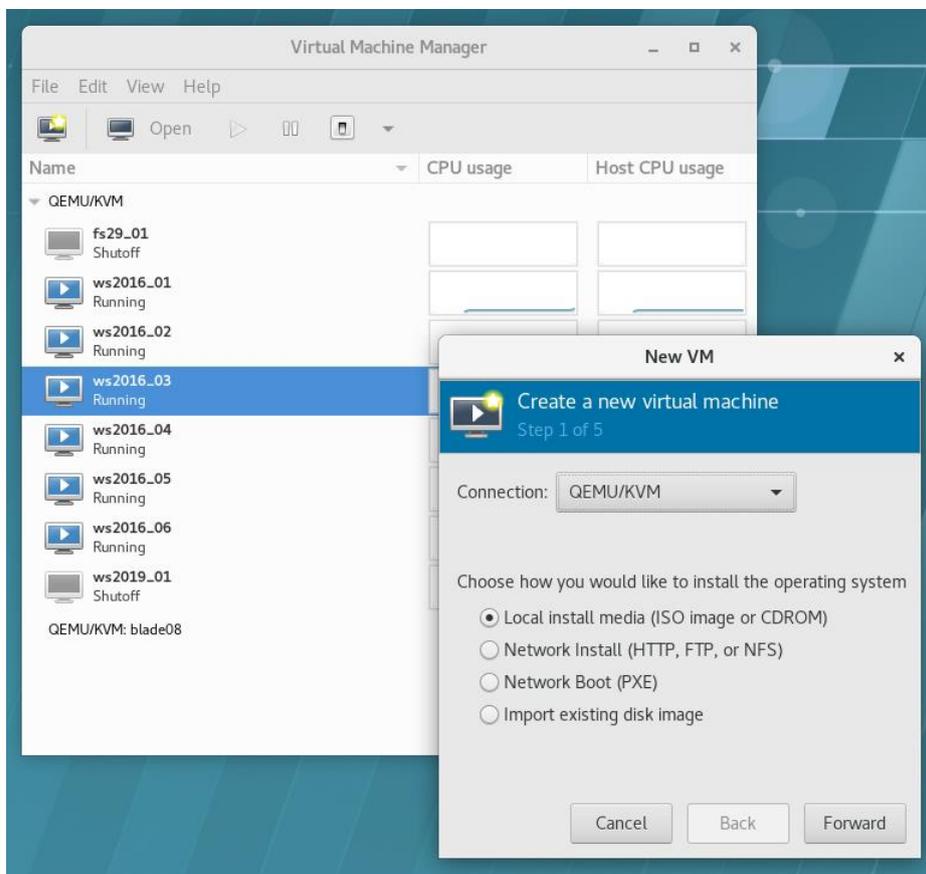
► Resource Ordering Preferences (1)

Fonte: Do autor (2019).

#### 4.2.1.4 Máquinas virtuais – *VirtualDomain*

Os últimos recursos configurados no cluster foram seis máquinas virtuais, executadas sob o *hypervisor* QEMU<sup>8</sup>/KVM, nativo do RHEL. Antes de incluir as VMs como recursos do cluster, elas foram criadas normalmente utilizando a interface gráfica do *Virtual Machine Manager*, a qual pode ser vista na Figura 27.

Figura 27 – Interface gráfica do *Virtual Machine Manager* com a tela de criação de uma nova VM



Fonte: Do autor (2019).

A distribuição de recursos de CPU e memória RAM do hospedeiro para as VMs, bem como o IP atribuído a cada uma, é demonstrada na Tabela 3.

<sup>8</sup> Emulador gratuito e livre que efetua virtualização de hardware, apresentando baixo desempenho se usado sozinho, mas, ao ser combinando com o KVM, possibilita executar máquinas virtuais a velocidades quase nativas do hospedeiro.

Tabela 3 – Distribuição dos recursos do hospedeiro entre as VMs

VM	Núcleos lógicos de CPU atribuídos	Memória RAM (GB)	IP
ws2016_01	4	16	192.168.150.21
ws2016_02	4	16	192.168.150.22
ws2016_03	4	16	192.168.150.23
ws2016_04	4	16	192.168.150.24
ws2016_05	8	32	192.168.150.25
ws2016_06	6	24	192.168.150.26

Fonte: Do autor (2019).

Para cada VM, foi criado um disco virtual de 40 GB, os quais foram alocados na pasta “/shared/vm-disks/” do armazenamento compartilhado GFS2, de forma a ficarem acessíveis por todos nós do cluster. A fim de atingir melhor desempenho, optou-se por usar discos virtuais no formato *img*, com forma de armazenamento *raw* e barramento iSCSI via *VirtIO*<sup>9</sup>.

Em todas as VMs, foi instalado o sistema operacional Windows Server 2016 Standard com interface gráfica habilitada. Para permitir o correto funcionamento do *VirtIO* para os drivers de disco e rede, bem como para melhor desempenho gráfico, foram instalados os drivers do *VirtIO* e o *qemu-guest-agent* em cada VM. Também foi ativado o acesso via *Remote Desktop Protocol* (RDP) para possibilitar o acesso remoto a elas.

Após a configuração inicial das VMs, estas foram adicionadas ao cluster existente como recursos deste. Estas passam a ser gerenciadas pelo Pacemaker através do agente de recursos “*ocf:heartbeat:VirtualDomain*”, que, por sua vez, se comunica com a API *libvirt* que gerencia o QEMU/KVM no RHEL.

Antes de criar os recursos de cada VM, foi necessário copiar o arquivo de configuração *Extensible Markup Language* (XML) de cada uma, localizado em “*etc/libvirt/qemu*”, para um local no armazenamento compartilhado – neste caso, para a pasta “/shared/vm-xmlls”. Feito isso, foi preciso desligar cada VM e indefinir elas do

<sup>9</sup> Padrão de virtualização de I/O (*Input/Output*) usado pelo *libvirt*, no qual os drivers da VM “sabem” que estão em um ambiente virtualizado e cooperam com o *hypervisor*, alcançando assim alto desempenho em operações de disco e rede (LIBVIRT, 2019).

*virsh*<sup>10</sup> através do comando “# *virsh* undefine ws2016\_0x”. Ao executar tal comando, o *virsh* deixa de gerenciar a máquina virtual, excluindo o arquivo XML do seu local original e desaparecendo do VMM.

Também foi necessário configurar chaves criptográficas de autenticação SSH entre os nós do cluster, para garantir a conexão entre estes nós, sem a necessidade do uso de senhas. Esta configuração é essencial para possibilitar a migração das máquinas virtuais em tempo real através de tunelamento SSH.

Por fim, foram criados os seis recursos, um para cada VM, intitulados “VM\_ws2016\_0n”, sendo o “n” o número referente a cada VM. Para isso, foi utilizado o comando abaixo, em que foram especificados o nome da VM, agente de recursos, arquivo de configurações, opções de migração, bem como *timeouts* para início, parada, migração e monitoramento.

```
# pcs resource create VM_ws2016_01 VirtualDomain
hypervisor="qemu:///system" config="/shared/vm-
xmls/ws2016_01.xml" migration_transport=ssh migrate_options="--
p2p --tunneled" op start timeout="120s" op stop timeout="120s"
op monitor timeout="30" interval="10" meta allow-migrate="true"
op migrate_from interval="0s" timeout="300s" op migrate_to
interval="0s" timeout="300s"
```

Em sequência, foram definidas regras de dependência para os recursos das VMs, especificando que estes só podem iniciar após o recurso de armazenamento compartilhado “clusterfs” iniciar. Caso uma VM inicie antes desse recurso, ela irá falhar, visto que seu disco virtual e arquivo de configuração estão armazenados no disco compartilhado. O comando utilizado para definir tal regra para o recurso “VM\_ws2016\_01” é apresentado abaixo.

```
# pcs constraint order start clusterfs-clone then VM_ws2016_01
```

---

<sup>10</sup> O *virsh* é um utilitário de linha de comando para gerenciamento de máquinas virtuais baseado na API *libvirt*, sendo o gerenciador padrão para o *hypervisor* KVM no RHEL (HERRMANN et al., 2018).

## 4.2.2 Panorama do cluster configurado

Finalizada a configuração do cluster, o seu monitoramento pode ser efetuado através do utilitário *crm\_mon*, que acompanha o Pacemaker. Através dele, é possível ver o estado em tempo real dos nós, quais recursos estão ativos e parados, possíveis erros, bem como qual é o nó tomador de decisões, chamado de *Designated Controller* (DC). O DC é eleito de forma automática e não tem grande significância para o administrador além do fato de seus logs conterem informações mais relevantes (BEEKHOF et al., 2017). A Figura 28 mostra o *crm\_mon* em execução no cluster configurado.

Figura 28 – Utilitário *crm\_mon* em execução

```

root@blade06:~
Stack: corosync
Current DC: blade08 (version 1.1.19-8.e17-c3c624ea3d) - partition with quorum
Last updated: Sun Jun  2 15:25:52 2019
Last change: Mon May 13 22:37:44 2019 by root via crm_resource on blade08

2 nodes configured
15 resources configured

Online: [ blade06 blade08 ]

Active resources:

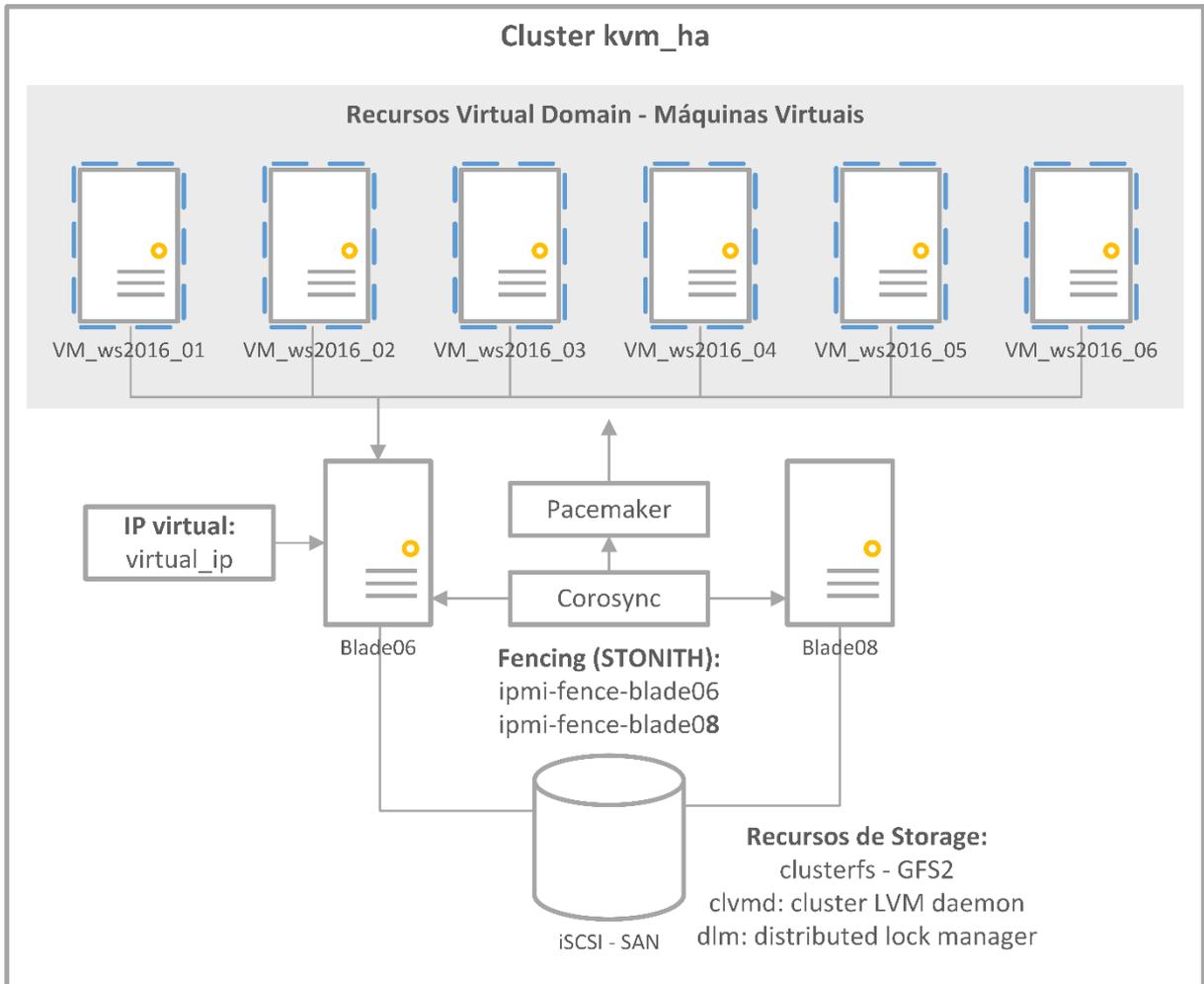
Clone Set: dlm-clone [dlm]
  Started: [ blade06 blade08 ]
Clone Set: clvmd-clone [clvmd]
  Started: [ blade06 blade08 ]
ipmi-fence-blade06      (stonith:fence_ipmilan):      Started blade06
ipmi-fence-blade08      (stonith:fence_ipmilan):      Started blade08
Clone Set: clusterfs-clone [clusterfs]
  Started: [ blade06 blade08 ]
virtual_ip              (ocf::heartbeat:IPaddr2):      Started blade06
VM_ws2016_01           (ocf::heartbeat:VirtualDomain): Started blade06
VM_ws2016_02           (ocf::heartbeat:VirtualDomain): Started blade06
VM_ws2016_03           (ocf::heartbeat:VirtualDomain): Started blade06
VM_ws2016_04           (ocf::heartbeat:VirtualDomain): Started blade06
VM_ws2016_05           (ocf::heartbeat:VirtualDomain): Started blade06
VM_ws2016_06           (ocf::heartbeat:VirtualDomain): Started blade06

```

Fonte: Do autor (2019).

O esquema da configuração final do cluster pode ser visto na Figura 29 abaixo, em que são ilustrados todos os recursos criados, bem como sua interligação para o perfeito funcionamento do ambiente.

Figura 29 – Esquema da configuração final do cluster "kvm\_ha"



Fonte: Do autor (2019).

## 5 EXPERIMENTOS E ANÁLISE DOS RESULTADOS

Neste capítulo, são relatados os experimentos efetuados no cluster a fim de demonstrar o seu consumo energético, bem como o gerenciamento de seus recursos em situações específicas. Junto a cada experimento, são mostrados os resultados obtidos e sua análise.

### 5.1 Experimento 1 – Tempos para inicialização do cluster e migração de recursos

O primeiro experimento teve por intuito mensurar os seguintes itens:

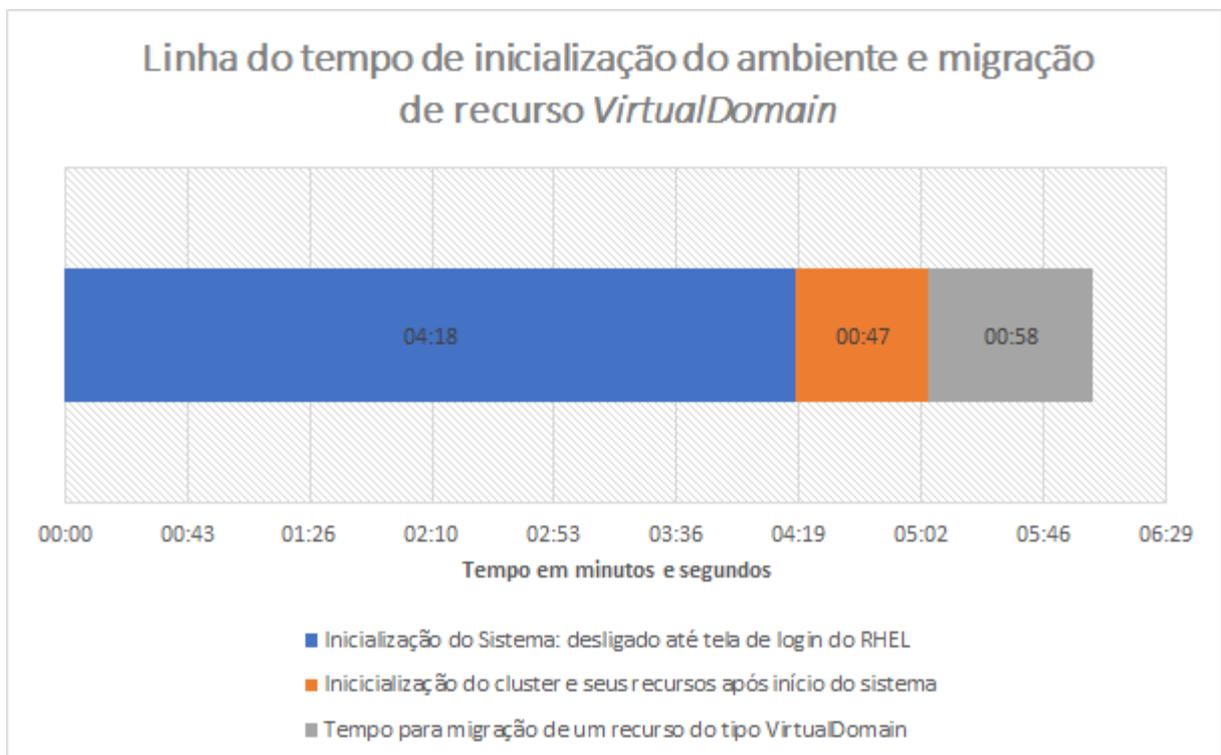
- a) Tempo de inicialização de cada blade, desde o momento que é pressionado o botão liga/desliga (ou enviado o comando “power on” via iDRAC) até a tela de login do RHEL;
- b) Tempo até o cluster e seus recursos iniciarem por completo, após inicialização do sistema, através do comando “# pcs cluster start –all” (este comando inicia os serviços do cluster em todos os nós);
- c) Tempo para um recurso do tipo *VirtualDomain* migrar de um nó para o outro, utilizando *live migration*, através do comando “# pcs resource move vm\_ws2016\_0x blade0x”;

- d) Tempo até todos os recursos migrarem e o cluster “estabilizar” após uma situação anormal/de emergência; neste caso, todos os recursos estavam em execução no nó *blade06*, o qual propositalmente foi desligado à força. O comportamento esperado era todos os recursos do cluster migrarem em tempo real para o nó restante, sendo restabelecidos com impacto mínimo ao usuário. Foi executado um teste com o *timeout* de migração definido em 120 segundos e outro em 300 segundos.

### 5.1.1 Resultados

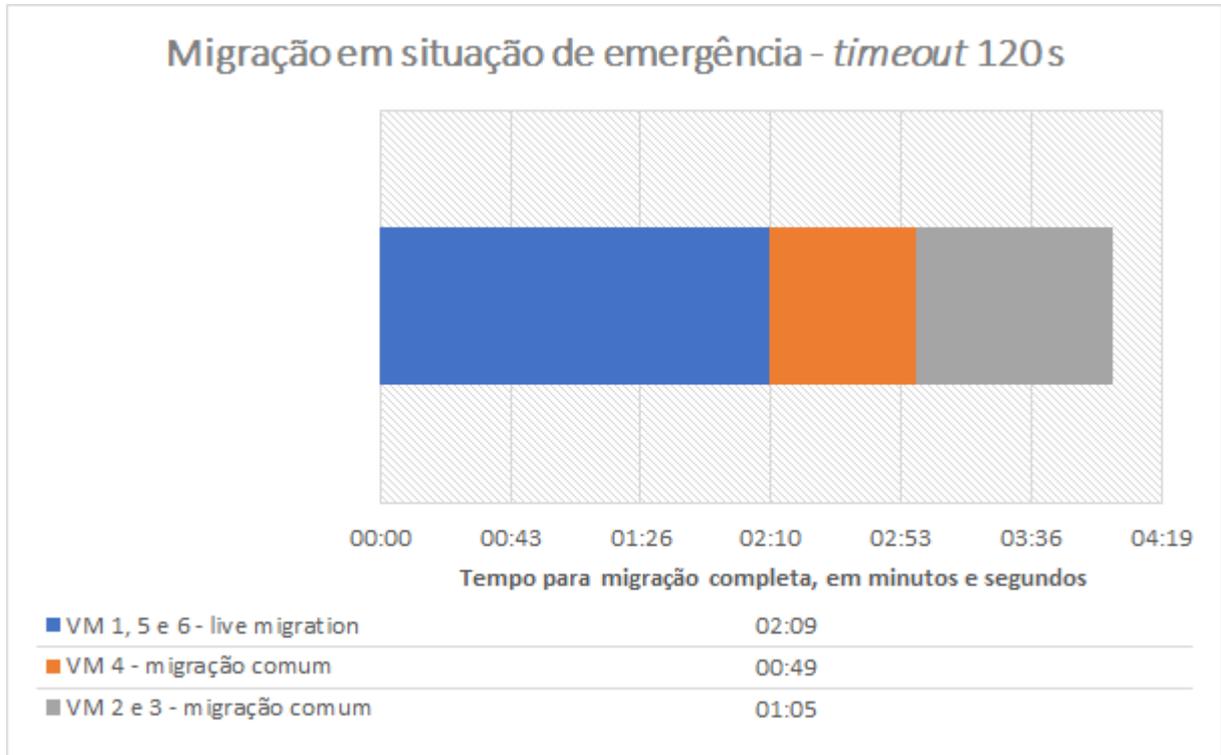
Os resultados dos itens ‘a’, ‘b’ e ‘c’ estão representados de forma combinada no Gráfico 3.

Gráfico 3 – Linha do tempo de inicialização do ambiente e migração de recurso *VirtualDomain*



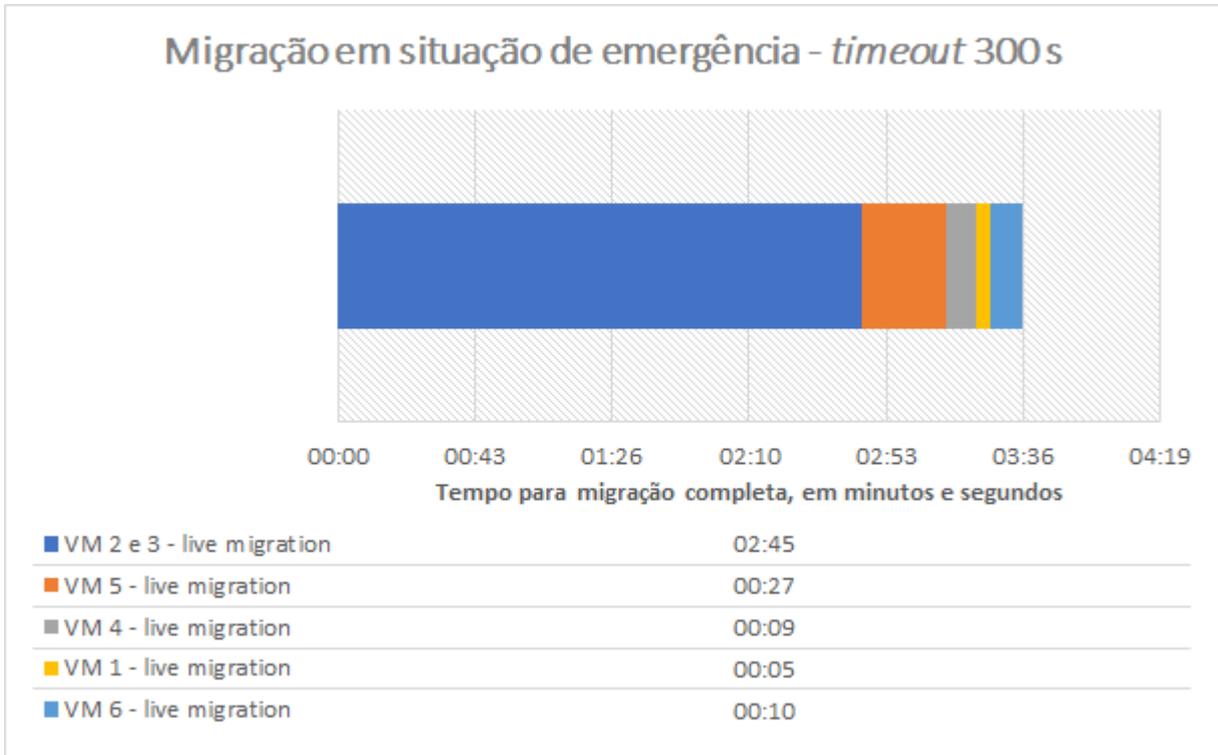
Fonte: Do autor (2019).

Já o resultado do item ‘d’ com *timeout* de migração de 120 segundos é apresentado no Gráfico 4.

Gráfico 4 – Migração em situação de emergência - *timeout* 120 s

Fonte: Do autor (2019).

O resultado do item 'd' com *timeout* de migração definido em 300 segundos é apresentado no Gráfico 5.

Gráfico 5 – Migração em situação de emergência - *timeout* 300 s

Fonte: Do autor (2019).

### 5.1.2 Análise dos resultados

Com base nos resultados observados a partir dos experimentos 'a' e 'b', pode-se verificar que o tempo total de inicialização do sistema, do cluster e seus recursos, até estes estarem disponíveis, pode ser considerado longo, visto os requisitos de disponibilidade e desempenho atuais. Grande parte desse tempo é devido à inicialização do servidor blade, que efetua diversas checagens durante o *boot*, aliado, depois, à inicialização do sistema operacional em si.

Com base no tempo total decorrido até a disponibilização dos recursos, é possível afirmar que a proposta de manter um dos nós desligado, não se mostra efetiva para um ambiente de alta disponibilidade. No caso de perda do nó ativo, o tempo para ligar o nó de *failover* e iniciar o cluster geraria um *downtime* elevado, bem como todos os recursos seriam reiniciados. No caso das VMs, isso poderia gerar perda de dados ou indisponibilidade de serviços.

Com ambos os nós ligados, observou-se que a migração controlada de um recurso do tipo *VirtualDomain* ocorre sem problemas, em menos de um minuto, conforme resultado do item 'c'. Nessa migração, não houve indisponibilidade do recurso, visto que esta ocorreu em tempo real (*live migration*).

Foi observado que, durante a migração, enquanto toda a memória RAM da VM estava sendo migrada, o uso de rede ficou em aproximadamente 697 Mbp/s, próximo ao limite da tecnologia de rede em uso. Presume-se que, com uma rede de maior velocidade, como uma de 10 Gbp/s, por exemplo, a migração ocorreria de forma mais rápida. Esse teste não foi possível dado limitações da infraestrutura do ambiente.

Em caso de pane em um dos nós, o cluster deve se reestruturar de forma invisível ao usuário, gerando o menor impacto possível. Os resultados do item 'd' demonstram os tempos de recuperação dos seis recursos do tipo *VirtualDomain* configurados, os quais, ao desligar o nó *blade06*, migraram para o nó *blade08*.

Inicialmente, na configuração destes recursos, havia sido definido o tempo de 120 segundos de *timeout* para as operações "*migrate\_to*" e "*migrate\_from*". Como ocorreu uma migração em massa, com grande quantidade de memória a ser reestabelecida, esse tempo se mostrou insuficiente para todos recursos migrarem, levando o Pacemaker a achar que a migração em tempo real falhou e reiniciar as VMs 4, 2 e 3.

Com base nesse resultado, o tempo de *timeout* para migração dos recursos de VMs foi aumentado para 300 segundos, e o teste, refeito. Com essa nova configuração, todas as VMs migraram em menos de quatro minutos, sem necessidade de reiniciá-las. Verificou-se, com esses testes, que o ambiente é capaz de prover alta disponibilidade de seus recursos e se reestruturar de forma automática em caso de pane em algum de seus nós.

Ainda baseado nos resultados dos testes, afirma-se que a configuração dos recursos é viável e funciona como o esperado, mas deve ser considerado o tempo de *timeout* no momento da parametrização desta. Estruturas com mais VMs irão exigir, conseqüentemente, tempos maiores de *timeout*, além de apresentarem maior tempo para recuperação total dos recursos.

## 5.2 Experimento 2 – Consumo de energia elétrica durante teste de stress com HeavyLoad

Este experimento teve o intuito de mensurar o consumo de energia elétrica do cluster sob diferentes cargas de trabalho e quando este se encontra em repouso. Foi medido o consumo em watt-segundo (Ws) de cada nó, a temperatura dos processadores e da placa-mãe, em quatro condições:

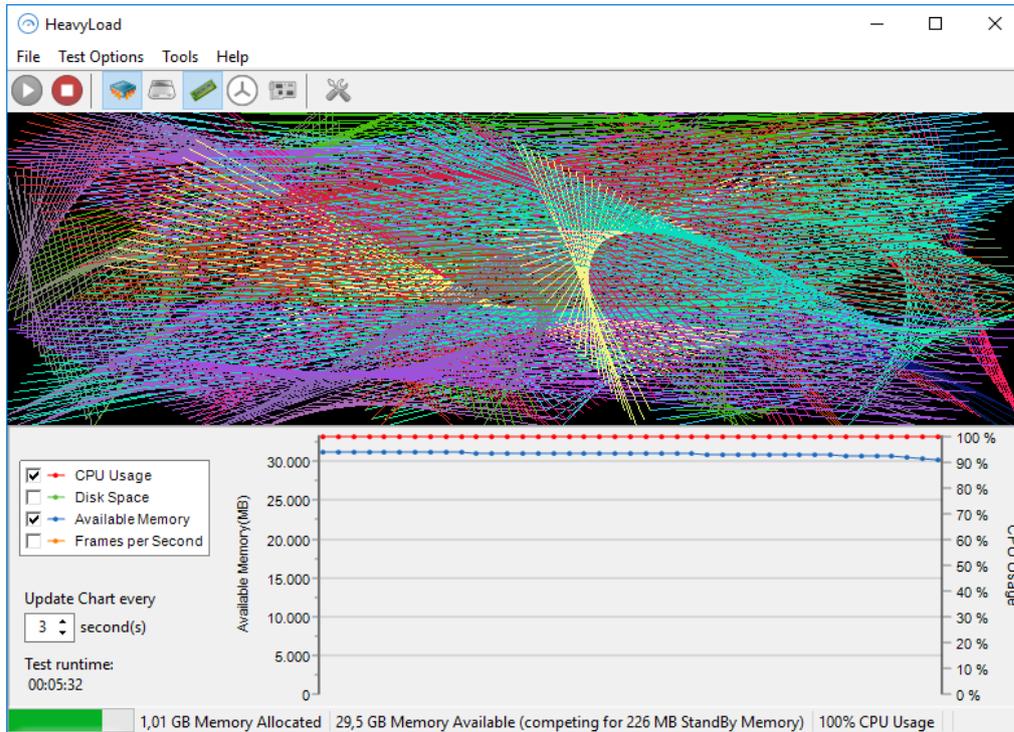
- a) cluster parado;
- b) cluster iniciado, mas com recursos sem carga;
- c) recursos de VM com carga máxima;
- d) VM 5 com carga máxima, e as VMs restantes ociosas.

A fim de simular cargas de trabalho nas VMs, foi instalado em cada uma o software HeavyLoad<sup>11</sup>. Ao executá-lo, este simulou uma carga de uso de 100% da CPU, ao mesmo tempo que continuamente alocou memória RAM até o limite de cada VM. A tela principal do software executando um teste na VM 5 pode ser vista na Figura 30.

---

<sup>11</sup> Software utilitário gratuito para executar testes de stress de CPU, disco, memória RAM e GPU, em sistemas Windows, através de técnicas diversas.

Figura 30 – Software HeavyLoad em execução na VM 5



Fonte: Do autor (2019).

As medidas de consumo e temperatura foram efetuadas através do iDRAC de cada servidor blade, o qual fornece dados em tempo real. Também foi mensurado o consumo do chassi de blades como um todo, através do CMC<sup>12</sup> (*Chassis Management Controller*). Dado que a estrutura é compartilhada, os dados do CMC incluem o consumo de servidores blades não participantes do cluster.

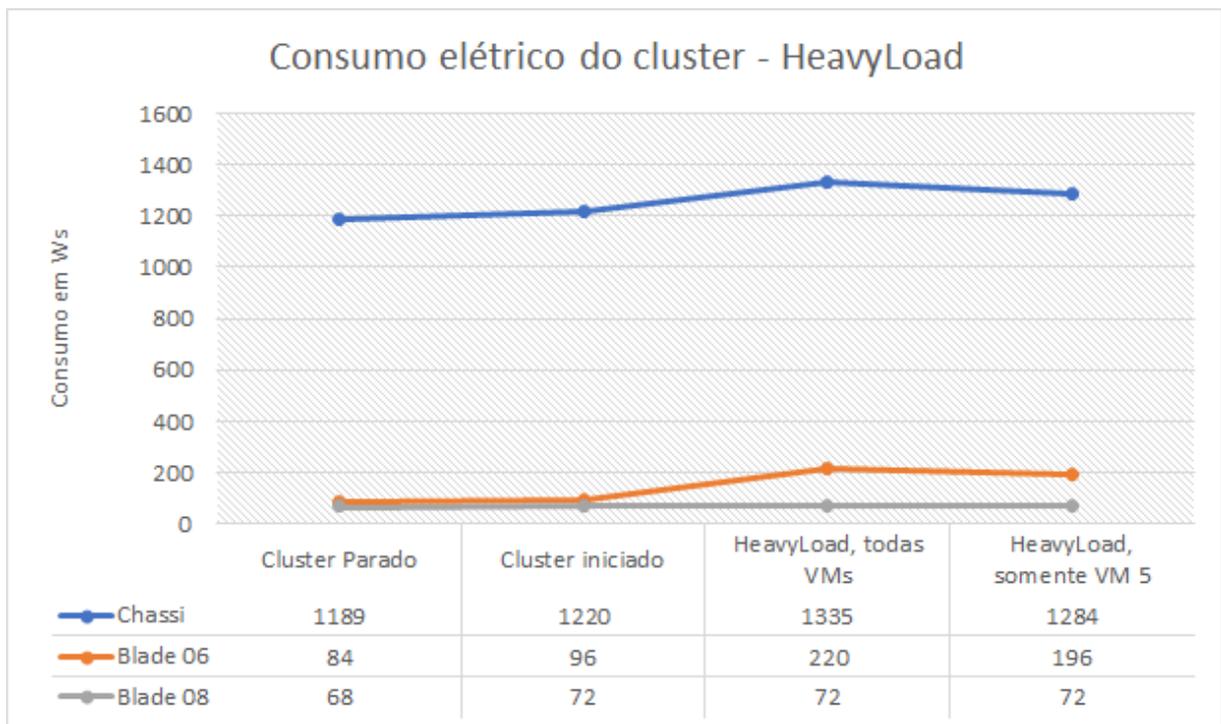
Não foi possível mensurar o consumo da estrutura através da utilização de um alicate amperímetro, por limitações da infraestrutura do rack e da sala no qual este estava localizado. Para mensurar com tal equipamento, seria necessário que os fios de alimentação do rack estivessem separados, e não unidos em um cabo. A sala também não contava com uma caixa de distribuição exclusiva, sendo a existente compartilhada com outros elementos do prédio, o que geraria muito ruído às medições.

<sup>12</sup> Semelhante ao iDRAC, o CMC fornece uma interface web para gerenciar e monitorar todo o receptáculo dos blades. Através dele, é possível ter controle sobre cada blade, ativos de rede, coolers, fontes, dentre outros elementos do chassi.

### 5.2.1 Resultados

O consumo elétrico em Ws dos nós do cluster, nas situações 'a', 'b', 'c' e 'd,' é mostrado no Gráfico 6. São demonstradas as medidas do CMC e de cada servidor blade.

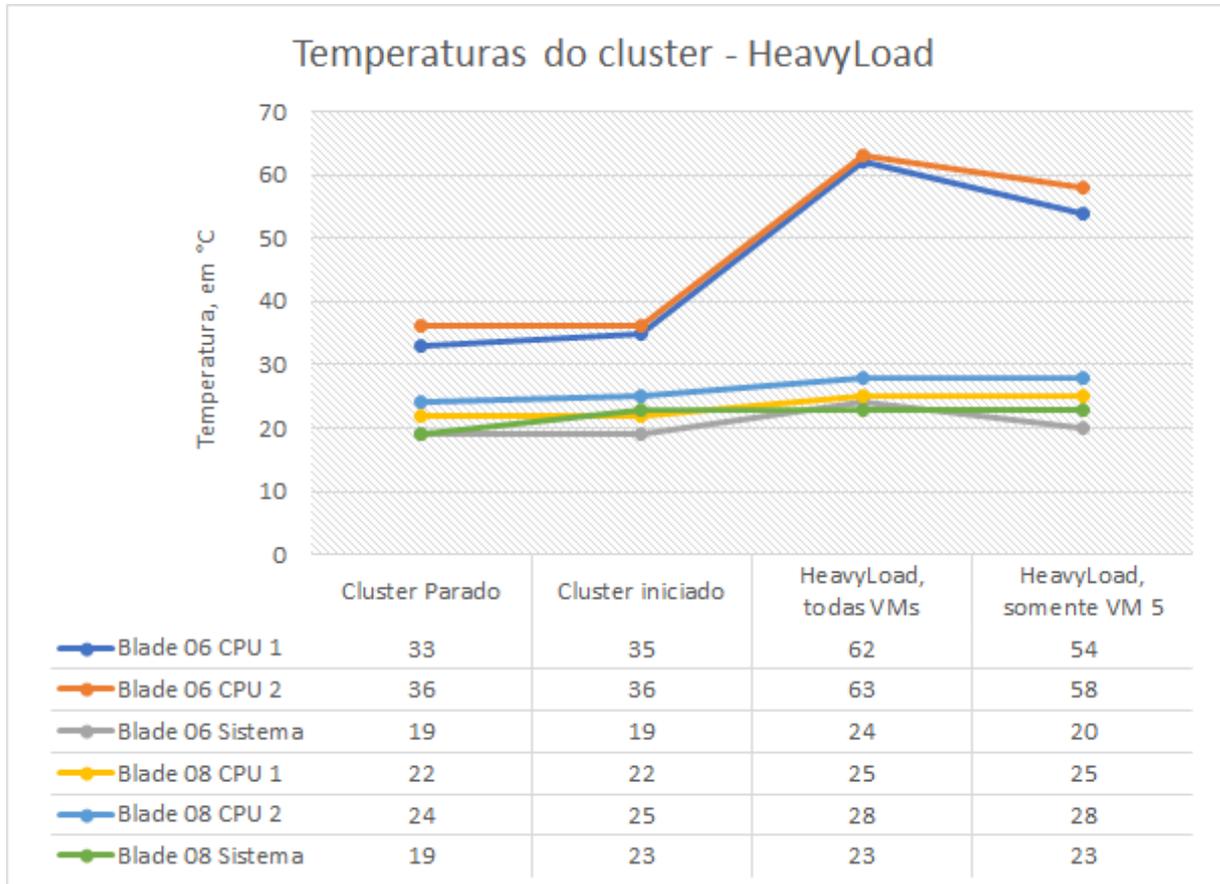
Gráfico 6 – Consumo elétrico do cluster com HeavyLoad



Fonte: Do autor (2019).

As temperaturas dos componentes dos nós do cluster, nas mesmas situações anteriormente citadas, são demonstradas no Gráfico 7. Foram mensuradas as temperaturas da CPU 1, CPU 2 e da placa-mãe.

Gráfico 7 – Temperaturas do cluster com HeavyLoad



Fonte: Do autor (2019).

### 5.2.2 Análise dos resultados

Através dos resultados obtidos, foi verificado que o cluster possui um relativo baixo consumo de energia quando é iniciado e os seus recursos estão ociosos. A diferença de consumo do cluster ativo para quando ele está totalmente parado é de apenas 12 Ws no nó *blade06* e 4 Ws no nó *blade08*. O maior consumo no primeiro nó se dá devido a este estar executando os recursos das VMs, além dos recursos de disco que executam em ambos os nós.

Com base nos resultados, verificou-se que as aplicações Pacemaker e Corosync, que gerenciam o cluster e seus recursos em si, apresentam baixo consumo, provendo um gerenciamento eficiente do ambiente.

Ao efetuar os testes de stress em todas as VMs, foi observado um grande aumento no consumo de energia no nó onde os recursos estavam em execução, sendo que este passou de 96 Ws para 220 Ws, chegando a atingir picos de 289 Ws. As temperaturas também aumentaram consideravelmente, mas permaneceram dentro dos limites aceitáveis pelo equipamento.

Ao executar a ferramenta *top*<sup>13</sup> no nó ativo, durante a execução do teste do item 'c', observou-se que quase todo o processamento do host foi utilizado, mantendo somente dois núcleos livres para o sistema operacional e gerenciamento do cluster em si. Esse comportamento era esperado, visto que foi planejado no momento do provisionamento de recursos às VMs. Na Figura 31, pode ser vista uma captura de tela da ferramenta *top* em execução no momento do teste.

Figura 31 – Ferramenta *top* em execução no momento do teste

```

root@blade06:~
top - 10:15:29 up 1 day, 23:19, 2 users, load average: 32.62, 32.83, 28.11
Tasks: 532 total, 7 running, 525 sleeping, 0 stopped, 0 zombie
%Cpu(s): 92.1 us, 2.3 sy, 0.0 ni, 5.6 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
KiB Mem : 13181203+total, 76087560 free, 52178032 used, 3546444 buff/cache
KiB Swap: 4194300 total, 3993084 free, 201216 used, 76791232 avail Mem

  PID USER      PR  NI  VIRT  RES  SHR  S  %CPU  %MEM    TIME+  COMMAND
 25886 qemu     20   0   33.0g 32.3g 11204 R 761.0 25.7 431:11.69 /usr/libexec/qemu-kvm -name ws2016_05 +
 26181 qemu     20   0   24.9g 24.2g 11164 S 589.0 19.3 313:38.40 /usr/libexec/qemu-kvm -name ws2016_06 +
 25909 qemu     20   0   16.9g 16.2g 11152 R 397.1 12.9 242:37.40 /usr/libexec/qemu-kvm -name ws2016_04 +
 25898 qemu     20   0   16.9g 16.3g 11112 R 394.8 12.9 286:39.45 /usr/libexec/qemu-kvm -name ws2016_03 +
 25926 qemu     20   0   16.9g 16.3g 11160 S 393.8 12.9 3071:32 /usr/libexec/qemu-kvm -name ws2016_02 +
 26198 qemu     20   0   16.9g 16.3g 11156 S 389.0 12.9 290:26.22 /usr/libexec/qemu-kvm -name ws2016_01 +
   192 root      25   5     0     0     0  R  12.3  0.0 115:31.99 [ksmd]
 11992 polkitd   20   0 630580 21232 5380 R  7.5  0.0 48:51.38 /usr/lib/polkit-1/polkitd --no-debug
 38186 root      20   0 162464 2916 1704 R  1.9  0.0  0:28.47 top
 11981 dbus      20   0 81448 4860 1996 S  1.6  0.0 12:47.94 /usr/bin/dbus-daemon --system --address+
 13279 root      20   0 1677956 24532 14212 S  1.6  0.0 22:19.75 /usr/sbin/libvirtd
 12059 root      20   0 396656 4620 3280 S  1.3  0.0  9:15.58 /usr/libexec/accounts-daemon
 24070 root      rt   0 244820 146556 114268 S  1.3  0.1 32:00.04 corosync
 24079 haclust+  20   0 113256 20140 10524 S  1.0  0.0 25:43.55 /usr/libexec/pacemaker/cib
   7758 arnhold+  20   0 456796 4048 2884 S  0.6  0.0  3:31.10 /usr/libexec/gsd-account
 44803 root      20   0 115832 2128 1404 S  0.6  0.0  0:00.02 /bin/sh /usr/lib/ocf/resource.d/heartb+
 11991 root      20   0 90500 3188 2332 S  0.3  0.0  0:30.77 /sbin/rngd -f
 13002 root      20   0 228436 11248 6440 S  0.3  0.0  3:43.32 /usr/sbin/snmpd -LS0-6d -f
 21053 root      20   0     0     0     0  S  0.3  0.0  0:04.54 [kworker/u96:1]
 24078 root      20   0 104264 8204 3036 S  0.3  0.0  0:10.20 /usr/sbin/pacemakerd -f
 24081 root      20   0 99912 5124 3996 S  0.3  0.0  2:36.97 /usr/libexec/pacemaker/lrmd

```

Fonte: Do autor (2019).

Durante a execução do teste descrito no item 'd', foi observado também um considerável aumento no consumo de energia no nó *blade06*, passando de 96 Ws para 196 Ws. Levando em consideração que a VM 5 é provisionada com 8 núcleos e

<sup>13</sup> Ferramenta presente em praticamente todas distribuições Linux, fornece uma interface via linha de comando a qual exibe, em tempo real, os processos ativos e detalhes destes, consumo de CPU, memória RAM, dentre outras informações úteis para o monitoramento do sistema.

32 GB de memória, 25 % dos recursos do nó hospedeiro, verificou-se que não há grande diferença no consumo de energia para quando as VMs estavam utilizando 90 % dos recursos do nó.

Por outro lado, o nó *blade08* mantém, durante os testes, o mesmo consumo de energia e temperaturas que apresenta enquanto o cluster está ocioso. Isso se dá devido a este nó não estar executando recursos de VMs, atuando somente como um nó de *failover*.

Uma das formas de otimizar o desempenho e, por fim, a eficiência energética do arranjo, seria distribuir as cargas de trabalho dos recursos do cluster entre os seus nós de forma automática. Este cenário não foi possível de ser implementado devido a limitações encontradas no Pacemaker e no agente de recursos *VirtualDomain*, bem como falta de ferramentas nativas que implementem isso em um ambiente do tipo configurado. De acordo com os dados analisados acima, essa configuração não se mostra vantajosa em um cluster com menos de três nós, visto que uma máquina trabalhando próximo a 100 % de sua capacidade apresenta o mesmo consumo de duas máquinas trabalhando a 50 %, além de perder os benefícios de redundância e alta disponibilidade.

O desenvolvimento de scripts para essa função seria uma alternativa, mas não foi abordado no escopo deste trabalho devido a sua complexidade e ao tempo disponível para a sua realização. Sistemas como Red Hat Virtualization (RHV), específico para virtualização, integram o balanceamento de carga de forma nativa, mas não provêm um ambiente de cluster como o configurado neste trabalho.

### **5.3 Experimento 3 – Consumo energético durante execução do Apache Benchmark**

O último experimento conduzido teve por objetivo observar o consumo de energia e temperaturas do cluster, durante a simulação de cargas de trabalho de servidores web nas VMs. A fim de prover tal cenário, foi instalado e configurado o

*Apache Web Server*<sup>14</sup> em cada VM, juntamente com uma página web simples localizada na própria máquina.

O Apache disponibiliza, juntamente com o software principal, uma ferramenta de benchmark chamada de Apache Benchmark (AB), a qual foi utilizada para simular as cargas de trabalho. O AB é capaz de simular um número customizável de acessos a uma página web, de forma concorrente ou não. A utilização da ferramenta é simplificada, exigindo somente três parâmetros para o teste, os quais estão descritos a seguir:

- '-n 1000000': representa o número de requisições a serem feitas, neste caso foram um milhão de requisições;
- '-n 50': representa o número de concorrências, ou seja, requisições simultâneas, neste caso foram cinquenta requisições concorrentes;
- 'localhost/': endereço da página web a ser testada, neste caso uma página configurada na própria máquina, com endereço local.

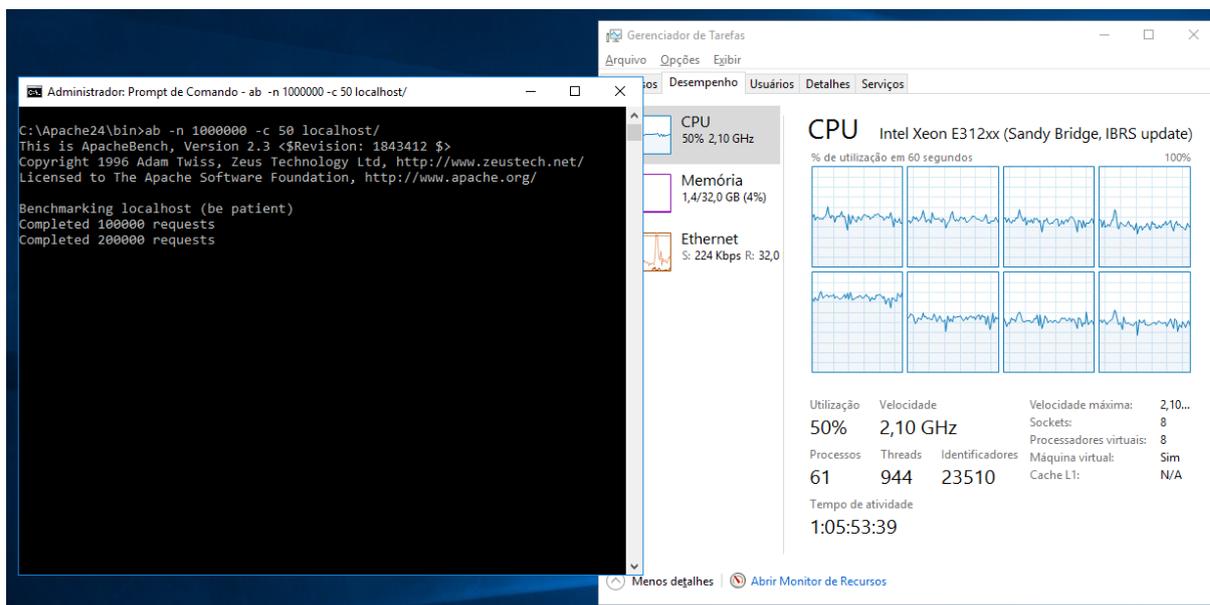
### 5.3.1 Resultados

O consumo de CPU em cada VM, durante o teste, teve relação com o número de núcleos provisionados a cada uma: nas VMs 1 a 4, com 4 núcleos cada, a utilização foi de aproximadamente 90 %; na VM 5, com 8 núcleos, o consumo foi de em torno de 50 %; na VM 6, com 6 núcleos, este consumo foi de aproximadamente 70 %. Na Figura 32, é exibido o AB em execução na VM 5, com o Gerenciador de Tarefas do Windows exibindo a utilização do processador.

---

<sup>14</sup> Servidor web livre do tipo HTTPD, criado em 1995, o qual suporta o protocolo HTTP 1.1. É um dos servidores web mais utilizados no mundo, sendo base de mais da metade dos sites existentes.

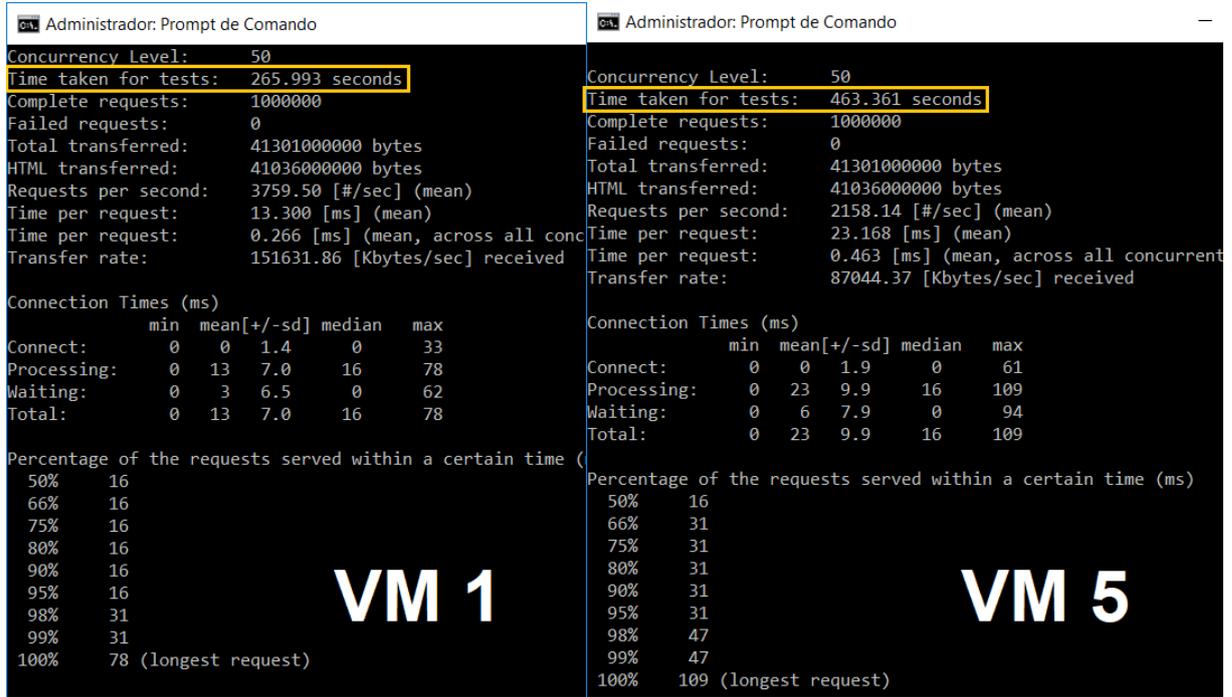
Figura 32 – AB em execução na VM 5, juntamente do Gerenciador de Tarefas



Fonte: Do autor (2019).

Da mesma forma, o tempo de execução do teste também variou de acordo com o poder de processamento disponível para cada VM, mas de forma inversa. Nas VMs com quatro núcleos, o teste foi executado mais rapidamente do que nas VMs com oito e seis. Dado este resultado, presume-se que o AB se comporta melhor em sistemas com menor número de núcleos, distribuindo de forma mais eficiente sua carga de trabalho. A diferença de tempo foi de aproximadamente 75 %, constatado a partir do item *Time taken for tests* dos relatórios finais do AB executado nas VM 1 e 5, estes apresentados na Figura 33.

Figura 33 – Relatórios finais do AB executado nas VMs 1 e 5



Fonte: Do autor (2019).

O AB foi executado com os mesmos parâmetros, em todas as VMs ao mesmo tempo, gerando aproximadamente 67 % de uso da CPU do nó *blade06*, conforme capturado na Figura 34, que apresenta a saída da ferramenta *top* no momento de execução dos testes.

Figura 34 – Saída da ferramenta *top* no momento de execução dos testes com AB

```

root@blade06:~
top - 20:35:38 up 1 day, 22:05, 1 user, load average: 18.80, 7.03, 2.96
Tasks: 453 total, 1 running, 452 sleeping, 0 stopped, 0 zombie
%Cpu(s): 67.3 us, 7.3 sy, 0.0 ni, 25.4 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
KiB Mem : 13181203+total, 11366604+free, 15057200 used, 3088796 buff/cache
KiB Swap: 4194300 total, 4194300 free, 0 used. 11394078+avail Mem

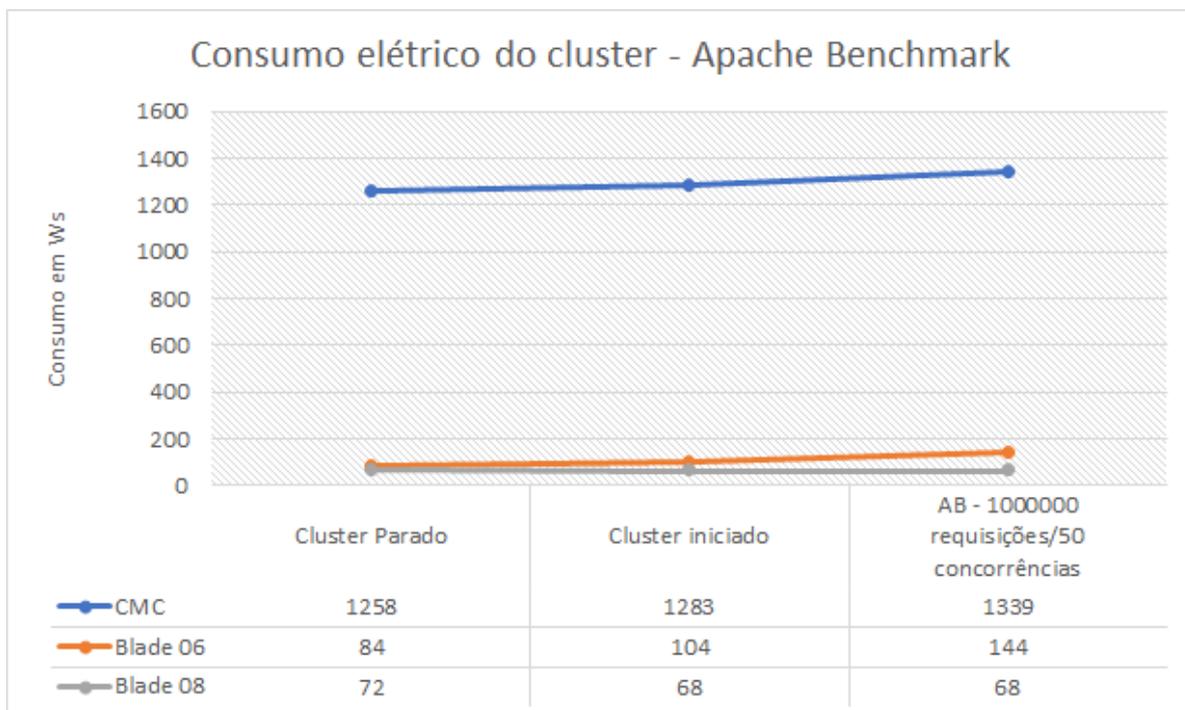
  PID USER      PR  NI   VIRT   RES   SHR  S  %CPU  %MEM    TIME+  COMMAND
 25624 qemu     20   0   32.9g   32.3g  11036 S  453.6  25.7  901:07.35 qemu-kvm
 25626 qemu     20   0   24.9g   24.3g  11032 S  430.6  19.3  568:19.80 qemu-kvm
 28331 qemu     20   0   16.9g   16.3g  11140 S  367.8  12.9  432:47.23 qemu-kvm
 30129 qemu     20   0   16.9g   16.3g  11144 S  366.1  12.9  435:20.17 qemu-kvm
 25630 qemu     20   0   16.9g   16.2g  11028 S  365.5  12.9  447:08.56 qemu-kvm
 33578 qemu     20   0   16.9g   16.3g  11148 S  364.5  12.9  427:46.74 qemu-kvm
   191 root      25   5     0     0     0  S    5.6  0.0   278:24.55 ksmd
 24459 root      rt   0  244776 146512 114268 S    3.3  0.1   31:01.65 corosync
 12044 root      20   0  906768 51036  8008  S    2.0  0.0    4:10.43 pcsd
   2852 root      20   0  162292  2620  1596  R    1.0  0.0    0:01.21 top
    20 root      rt   0     0     0     0  S    0.3  0.0    0:00.51 migration/2
    45 root      rt   0     0     0     0  S    0.3  0.0    0:00.92 migration/7
    60 root      rt   0     0     0     0  S    0.3  0.0    0:00.42 migration/+
 13057 root      20   0  573932 19156  6020  S    0.3  0.0    0:20.40 tuned
 13063 root      20   0  228432 11248  6440  S    0.3  0.0    6:00.43 snmpd
 13237 root      20   0  426084 24928  6752  S    0.3  0.0    1:49.60 fail2ban-s+
 24471 haclust+ 20   0  128028  7060  3856  S    0.3  0.0    0:14.79 attrd

```

Fonte: Do autor (2019).

O consumo elétrico em Ws dos nós do cluster durante a execução dos testes com o AB é mostrado no Gráfico 8. São explicitadas as medidas do CMC e de cada servidor blade, comparadas ao cluster parado e ao cluster iniciado com seus recursos ociosos.

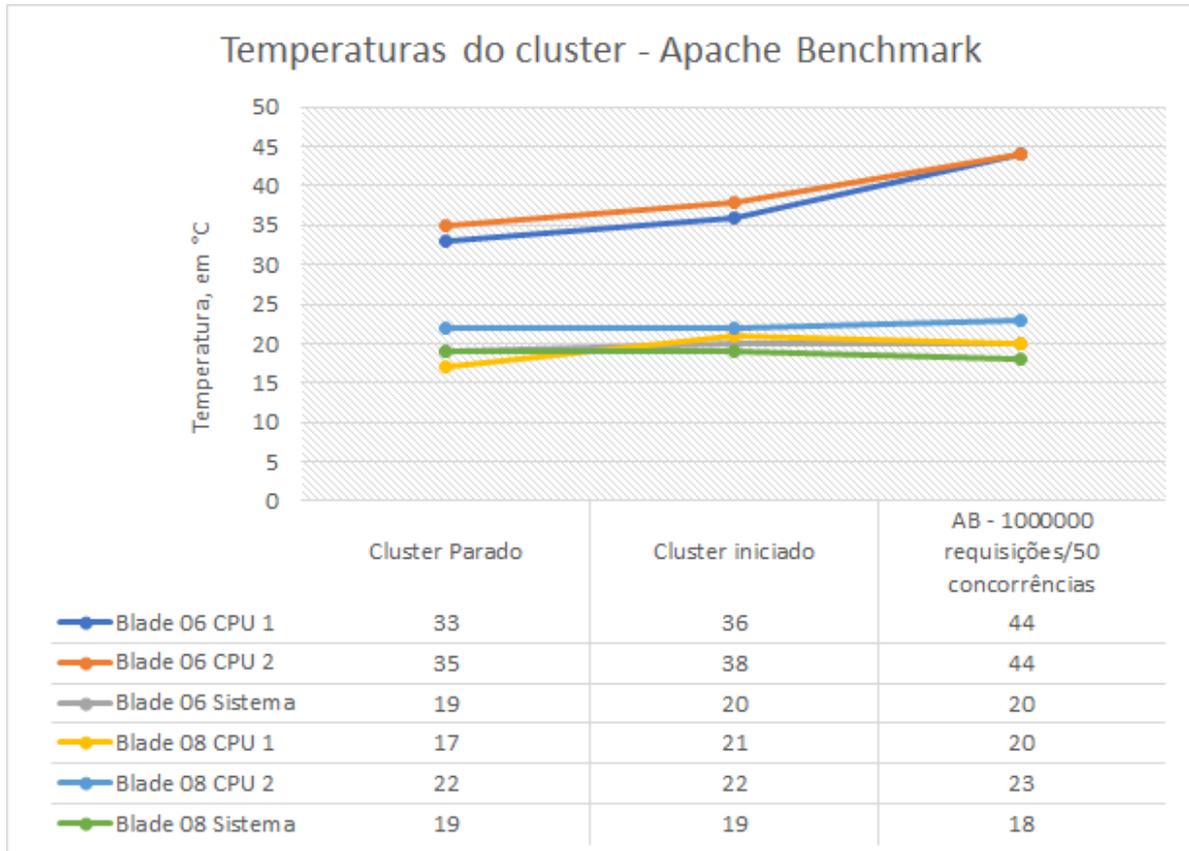
Gráfico 8 – Consumo elétrico do cluster com Apache Benchmark



Fonte: Do autor (2019).

As temperaturas dos componentes dos nós do cluster, nas mesmas situações anteriormente citadas, são mostradas no Gráfico 9. Foram mensuradas as temperaturas da CPU 1, CPU 2 e da placa-mãe.

Gráfico 9 – Temperaturas do cluster com Apache Benchmark



Fonte: Do autor (2019).

### 5.3.2 Análise dos resultados

Ao analisar os resultados obtidos nos testes com o AB foi verificado que, mesmo com o um uso geral maior de CPU do nó hospedeiro, o consumo elétrico se mostra menor do que os testes com o HeavyLoad sendo executado somente na VM 5. O uso de CPU do nó *blade06* foi de em torno de 67 %, enquanto que com o HeavyLoad na VM 5 este número foi de somente 25 %. O consumo ficou em 144 Ws, um aumento de 40 Ws, tendo como referência o cluster com seus recursos ociosos.

Nestes testes, é importante salientar que, mesmo com alto consumo de CPU como um todo, nenhum dos 32 núcleos estava 100 % em uso, havendo melhor distribuição de carga dentro das VMs. Os testes com o AB também não exigiram tanta memória RAM como os testes de stress realizados com o HeavyLoad, o que também contribuiu com o menor consumo de energia do ambiente.

Com base no resultado obtido, verificou-se que o cluster se mostra mais eficiente energeticamente em situações onde o processamento é mais bem distribuído, e não concentrado totalmente em poucos núcleos, como o caso do teste com o HeavyLoad na VM 5.

Observa-se também que em uma aplicação regular, assim como o Apache, a carga de utilização dos recursos mantém-se em níveis médios, assim como o consumo de energia elétrica. Isto demonstra que o ambiente se apresenta eficiente, em um cenário mais próximo de um caso de uso real, como é caso de um servidor web, onde os recursos tenham sido provisionados adequadamente.

#### **5.4 Consumo de energia elétrica do chassi de blades**

O chassi de blades utilizado possui seis fontes de alimentação de 2700 W cada, sendo três delas alimentadas pela PDU 1 e as outras três pela PDU 2. Esta configuração provê redundância em caso de falha de uma ou mais fontes ou de uma das PDUs.

Foi verificado no CMC que todas as seis fontes estavam ativas, mesmo com carga mínima, gerando consumo desnecessário de energia elétrica, visto que mesmo sem uso as fontes consomem energia se estiverem ligadas. O consumo estimado de todas as fontes ativas, com esta configuração, era de 1403,45 Ws. A listagem de fontes do chassi é exibida na Figura 35, junto com a corrente e voltagem demandada por cada.

Figura 35 – Listagem de fontes do chassi apresentada no CMC

The screenshot shows the Dell Chassis Management Controller (CMC) interface. The top navigation bar includes 'Properties', 'Setup', 'Power', 'Logs', 'Network', 'User Authentication', 'Alerts', 'Troubleshooting', 'Update', and 'Security'. The 'Power' tab is selected, and the 'Budget Status' sub-tab is active. The main content area displays two tables:

**Server Modules**

Slot	Name	Module Type	Priority	Power State	Budget Allocation	
					Minimum	Actual
1	b2lamina1	PowerEdgeM710HD	1	On	358 W	369 W
2	b2lamina2.univates.br	PowerEdgeM710HD	1	On	358 W	369 W
3	b2lamina3.	PowerEdgeM710HD	1	On	358 W	369 W
4	b2lamina4	PowerEdgeM710HD	1	On	358 W	369 W
5	b2lamina5	PowerEdgeM710HD	1	On	346 W	358 W
6	blade06	PowerEdge M520	1	On	294 W	330 W
7	b2lamina7	PowerEdge M520	1	On	299 W	317 W
8	blade08	PowerEdge M520	1	On	264 W	281 W

**Chassis Power Supplies**

Name	Power State	Input Volts	Input Current	Output Rated Power
PS-1	Online	230.5 V	0.9 A	2700 W
PS-2	Online	224.5 V	0.6 A	2700 W
PS-3	Online	230.0 V	0.9 A	2700 W
PS-4	Online	225.0 V	1.0 A	2700 W
PS-5	Online	224.5 V	1.4 A	2700 W
PS-6	Online	225.0 V	1.4 A	2700 W

Fonte: Do autor (2019).

Dessa forma, com o intuito de diminuir o consumo de energia elétrica do arranjo, foi ativada o atributo *Enable Dynamic Power Supply Engagement* nas configurações de energia do CMC. Esse atributo analisa o consumo demandado pelos blades do chassi e mantém ativas somente as fontes necessárias para a potência demandada por elas, colocando as outras em modo de espera.

O atributo *Redundant Policy* foi definido para *Power Supply Redundancy*, o que levou uma fonte adicional a também ficar ativa, a fim de prover redundância em caso de falha de uma das outras em uso. Essa configuração levou o CMC a colocar em espera três fontes, como é possível verificar na Figura 36, diminuindo o consumo total das fontes para 1301,05 Ws.

Figura 36 – Listagem de fontes do chassi exibidas no CMC

The screenshot shows the Dell Chassis Management Controller (CMC) interface for a PowerEdge M1000e chassis. The 'Power' tab is selected, and the 'Budget Status' sub-tab is active. The 'Standby Input Power Capacity' is shown as 8916 W. The 'Server Modules' table lists 8 modules with their respective power states and budget allocations. The 'Chassis Power Supplies' table lists 6 power supplies with their current states and output ratings.

Slot	Name	Module Type	Priority	Power State	Minimum	Actual
1	b2lamina1	PowerEdgeM710HD	1	On	358 W	369 W
2	b2lamina2.univates.br	PowerEdgeM710HD	1	On	358 W	369 W
3	b2lamina3.	PowerEdgeM710HD	1	On	358 W	369 W
4	b2lamina4	PowerEdgeM710HD	1	On	358 W	369 W
5	b2lamina5	PowerEdgeM710HD	1	On	346 W	358 W
6	blade06	PowerEdge M520	1	On	294 W	330 W
7	b2lamina7	PowerEdge M520	1	On	299 W	317 W
8	blade08	PowerEdge M520	1	On	264 W	281 W

Name	Power State	Input Volts	Input Current	Output Rated Power
PS-1	Online	226.0 V	1.8 A	2700 W
PS-2	Online	220.0 V	1.7 A	2700 W
PS-3	Online	225.5 V	2.0 A	2700 W
PS-4	Standby	231.0 V	0.1 A	2700 W
PS-5	Standby	229.5 V	0.1 A	2700 W
PS-6	Standby	232.0 V	0.1 A	2700 W

Fonte: Do autor (2019).

Com base no resultado, observa-se que o consumo total de energia elétrica do chassi foi reduzido em aproximadamente 100 Ws, valor equiparado ao consumo de um blade ocioso. Além da redução do consumo, ao colocar parte das fontes em espera presume-se um aumento de vida útil das fontes do chassi, diminuindo o seu desgaste provocado pelo uso contínuo. Uma medição de maior acurácia, com um amperímetro alicate, não foi possível devido a limitações da infraestrutura do rack, conforme citado no 4º parágrafo de 5.2.

## 6 CONSIDERAÇÕES FINAIS

Este trabalho teve como proposta configurar, analisar e aprimorar a eficiência energética de um cluster de alta disponibilidade baseado no RHEL 7.6, provendo como recursos máquinas virtuais executadas sob o *hypervisor* KVM, em um ambiente de alto desempenho. Ao final, foi possível evidenciar que grande parte dos objetivos foram atingidos.

A configuração do cluster se mostrou desafiadora e mais demorada do que inicialmente previsto, dadas as diversas particularidades e etapas detalhadas ao longo deste trabalho que são necessárias para a correta configuração de um ambiente do tipo. Ao final da configuração, através da simulação de situações de pane e recuperação, evidenciou-se a sua capacidade de prover máquinas virtuais como recursos de alta disponibilidade, de forma transparente ao usuário.

As variações do consumo de energia elétrica do cluster foram evidenciadas através de outros dois experimentos, a partir dos quais constatou-se que o cluster apresenta maior eficiência energética quando apresentado a uma carga de trabalho mais bem distribuída, onde nenhum dos núcleos individuais atinja 100 % de uso. Concluiu-se também que o nó de *failover* apresenta um baixo, mas perceptível consumo enquanto ocioso, o qual não foi possível diminuir sem prejudicar a disponibilidade dos recursos.

Um dos objetivos propostos era diminuir o consumo de energia elétrica do cluster e, conseqüentemente, aumentar sua eficiência energética, através do uso de técnicas de balanceamento automático de carga. Este objetivo não foi atingido, devido

à falta de ferramentas que possibilitassem a implementação desta técnica, em recursos do tipo *VirtualDomain* configurados no Pacemaker, sem comprometer o ambiente de alta disponibilidade.

Foi possível evidenciar a redução de consumo de energia elétrica do chassi, que abriga os blades do cluster, após a ativação do atributo *Enable Dynamic Power Supply Engagement* presente no CMC. Por meio do gerenciamento mais eficiente das fontes de alimentação, além da redução no consumo, presume-se que esta configuração contribua com o aumento de vida útil dessas, beneficiando todos os usuários do chassi e não somente os do cluster configurado.

Como contribuição, este trabalho apresenta um passo-a-passo detalhado para uma possível configuração de um ambiente do mesmo tipo, na própria instituição de ensino ou em outros locais, seja para fins educativos ou profissionais. Foram detalhadas peculiaridades cruciais na configuração, que são de grande valia no momento da criação e parametrização de um arranjo deste tipo.

Outra contribuição é apresentação do KVM: como uma alternativa gratuita e viável para virtualização de sistemas operacionais em ambientes de alto desempenho; do Corosync e Pacemaker: como ferramentas eficientes no gerenciamento de um cluster de alta disponibilidade; e do RHEL 7.6: como um sistema operacional abrangente, customizável e de alto desempenho. Ferramentas estas não tão difundidas ainda na região, mas que, com uma parametrização adequada, se mostram poderosas e flexíveis o bastante para os mais variados tipos de infraestrutura.

Como sugestão de trabalhos futuros, cita-se o desenvolvimento de um possível *script* ou ferramenta capaz de efetuar o balanceamento de carga de recursos do tipo *VirtualDomain*, em um cluster de alta disponibilidade gerenciado pelo Corosync e Pacemaker, com o intuito de utilizar de forma mais eficiente os recursos disponíveis. Ainda aliado a isso, cita-se também a configuração de um cluster onde o nó de *failover* ficasse desligado, ou em estado de suspensão com consumo energético próximo de zero, sendo somente ativado caso necessário.

Por fim, outra sugestão seria utilizar o sistema Red Hat Virtualization, o qual provê ferramentas para virtualização e balanceamento de carga, mas trabalha de forma diferente ao de um cluster de alta disponibilidade.

## REFERÊNCIAS

ALI, Syed. **Practical Linux Infrastructure**. [S. l.]: Apress, 2014. 320 p. ISBN: 978-1-4842-0511-2. E-book. Disponível em: <<https://books.google.com.br/books?id=VVwnCgAAQBAJ&printsec=frontcover&hl=pt-BR#v=onepage&q&f=false>>. Acesso em: 19 out. 2018.

BAKER, Mark. **Cluster Computing White Paper**. Portsmouth: University of Portsmouth, 2000. Disponível em: <<https://arxiv.org/ftp/cs/papers/0004/0004014.pdf>>. Acesso em: 17 out. 2018.

BEEKHOF, Andrew et al. **Pacemaker 1.1 Configuration Explained: An A-Z guide to Pacemaker's Configuration Options**. [S.l.]: Clusterlabs, 2017. E-book. Disponível em: <[https://clusterlabs.org/pacemaker/doc/en-US/Pacemaker/1.1/html/Pacemaker\\_Explained/index.html](https://clusterlabs.org/pacemaker/doc/en-US/Pacemaker/1.1/html/Pacemaker_Explained/index.html)>. Acesso em: 29 mai. 2019.

BUYYA, Rajkumar. PARMON: a portable and scalable monitoring system for clusters. **Software: Practice and Experience**, [S. l.], v. 30, n. 7, p. 723-739, 2000. Disponível em: <<http://www.buyya.com/papers/parmon.pdf>>. Acesso em: 17 out. 2018.

CALWELL, Chris; OSTENDORP, Peter. 80 plus: A strategy for reducing the inherent environmental impacts of computers. In: 2005 IEEE INTERNATIONAL SYMPOSIUM ON ELECTRONICS AND THE ENVIRONMENT, 2005, New Orleans, Luisiana. **Proceedings...** New Orleans, Luisiana: IEEE, 2005, p. 151-156. Disponível em: <<https://www.computer.org/csdl/proceedings/isee/2005/8910/00/01437012.pdf>>. Acesso em: 21 out. 2018.

CARISSIMI, Alexandre. Virtualização: da teoria a soluções. In: 26º SIMPÓSIO BRASILEIRO DE REDES DE COMPUTADORES E SISTEMAS DISTRIBUÍDOS–SBRC, 2008, Rio de Janeiro. **Livro Texto dos Minicursos...** Rio de Janeiro: Sociedade Brasileira de Computação, 2008, p. 173-207. Disponível em: <<http://www.jvasconcellos.com.br/unijorge/wp-content/uploads/2012/01/cap4-v2.pdf>>. Acesso em: 19 out. 2018.

CERVO, Amado L.; BERVIAN, Pedro A.; DA SILVA, Roberto. **Metodologia Científica**. 6. Ed. São Paulo: Pearson Prentice Hall: 2007.

COSTA, Hebert Luiz Amaral. **Alta disponibilidade e balanceamento de carga para melhoria de sistemas computacionais críticos usando software livre**: um estudo de caso. 2009. Dissertação (Pós-Graduação em Ciência da Computação) – Universidade Federal de Viçosa, Minas Gerais, 2009. Disponível em: <[http://www.dpi.ufv.br/arquivos/ppgcc/dissertacoes/2009-ms-Hebert\\_Luiz\\_Amaral\\_Costa.pdf](http://www.dpi.ufv.br/arquivos/ppgcc/dissertacoes/2009-ms-Hebert_Luiz_Amaral_Costa.pdf)>. Acesso em: 06 jul. 2019.

COULOURIS, George et al. **Distributed Systems Concepts and Design**. 5 ed. Boston: Pearson, 2012. 1067 p. ISBN: 978-0-13-214301-1. E-book. Disponível em: <<http://www.gecg.in/papers/ds5thedn.pdf>>. Acesso em: 20 set. 2018.

DALFOVO, Michael Samir; LANA, Rogério Adilson; SILVEIRA, Amélia. Métodos quantitativos e qualitativos: um resgate teórico. **Revista Interdisciplinar Científica Aplicada**, Blumenau, v. 2, n. 4, p. 01-13, Sem II. 2008. Disponível em: <[http://www.aedmoodle.ufpa.br/pluginfile.php/168069/mod\\_forum/attachment/271244/MONOGRAFIAS%20M%C3%89TODOS%20QUANTITATIVOS%20E%20QUALITATIVOS.pdf](http://www.aedmoodle.ufpa.br/pluginfile.php/168069/mod_forum/attachment/271244/MONOGRAFIAS%20M%C3%89TODOS%20QUANTITATIVOS%20E%20QUALITATIVOS.pdf)>. Acesso em: 23 out. 2018.

DANTAS, Mario. **Computação Distribuída de Alto Desempenho**: Redes, Clusters e Grids Computacionais. Rio de Janeiro: Axcel Books do Brasil Editora, 2005. 275 p. ISBN: 85-7323-240-4.

DELL. **Dell Lifecycle Controller GUI v2.30.30.30 User's Guide**. Rev. A00. [S.I.]: Dell, 2016. E-book. Disponível em: <[https://topics-cdn.dell.com/pdf/idrac7-8-lifecycle-controller-v2.30.30.30\\_users-guide2\\_en-us.pdf](https://topics-cdn.dell.com/pdf/idrac7-8-lifecycle-controller-v2.30.30.30_users-guide2_en-us.pdf)>. Acesso em: 26 mai. 2019.

DELL. **Integrated Dell Remote Access Controller 8/7 Version 2.60.60.60 User's Guide**. Rev. A00. [S.I.]: Dell, 2018. E-book. Disponível em: <[https://topics-cdn.dell.com/pdf/idrac7-8-lifecycle-controller-v2606060\\_users-guide\\_en-us.pdf](https://topics-cdn.dell.com/pdf/idrac7-8-lifecycle-controller-v2606060_users-guide_en-us.pdf)>. Acesso em: 26 mai. 2019.

DOLEŽELOVÁ, Marie et al. **Red Hat Enterprise Linux 7 Power Management Guide**. [Raleigh]: Red Hat, 2018. E-book. Disponível em: <[https://access.redhat.com/documentation/en-us/red\\_hat\\_enterprise\\_linux/7/pdf/power\\_management\\_guide/Red\\_Hat\\_Enterprise\\_Linux-7-Power\\_Management\\_Guide-en-US.pdf](https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/7/pdf/power_management_guide/Red_Hat_Enterprise_Linux-7-Power_Management_Guide-en-US.pdf)>. Acesso em: 10 out. 2018.

DUBINSKI, John et al. High Performance Commodity Networking in a 512-CPU Teraflop Beowulf Cluster for Computational Astrophysics. In: SC2003, 2003, Phoenix, Arizona. **Paper...** Phoenix, Arizona, 2003, 11 p. Disponível em: <<https://arxiv.org/pdf/astro-ph/0305109.pdf>>. Acesso em: 10 out. 2018.

FELLER, Eugen; MORIN, Christine; LEPRINCE, Daniel. State of the Art of Power Saving in Clusters and Results from the EDF Case Study. **Rapports de recherche**, [S. I.], v. 2, n. 7473, p. 1-39, dez. 2010. Disponível em: <[https://www.researchgate.net/profile/Christine\\_Morin/publication/228584181\\_State\\_](https://www.researchgate.net/profile/Christine_Morin/publication/228584181_State_)

of\_the\_Art\_of\_Power\_Saving\_in\_Clusters\_and\_Results\_from\_the\_EDF\_Case\_Study/links/0046353b1534f57dfd000000/State-of-the-Art-of-Power-Saving-in-Clusters-and-Results-from-the-EDF-Case-Study.pdf>. Acesso em: 06 out. 2018.

FENG, Wu-chun; CAMERON, Kirk. The green500 list: Encouraging sustainable supercomputing. **Computer**, [S. l.], v. 40, n. 12, 2007. Disponível em: <<http://synergy.cs.vt.edu/pubs/papers/feng-ieeeec-g500list.pdf>>. Acesso em: 21 out. 2018.

FENG, Wu-chun; FENG, Xizhou; GE, Rong. Green supercomputing comes of age. **IT Professional**, [S. l.], n. 1, p. 17-23, fev. 2008. Disponível em: <[https://www.researchgate.net/profile/Xizhou\\_Feng/publication/3426961\\_Green\\_Supercomputing\\_Comes\\_of\\_Age/links/55afb07708ae32092e06d2ef/Green-Supercomputing-Comes-of-Age.pdf](https://www.researchgate.net/profile/Xizhou_Feng/publication/3426961_Green_Supercomputing_Comes_of_Age/links/55afb07708ae32092e06d2ef/Green-Supercomputing-Comes-of-Age.pdf)> Acesso em: 30 set. 2018.

FLYNN, Michael J. Some computer organizations and their effectiveness. **IEEE Transactions on Computers**, [S. l.], v. C-21, n. 9, p. 948-960, set. 1972. Disponível em: <<https://sites.google.com/site/arquiteturaiesgo/artigos/Some%20computer%20organizations%20and%20their%20effectiveness.pdf>>. Acesso em: 01 out. 2018.

FOSTER, Ian et al. Cloud computing and grid computing 360-degree compared. In: GRID COMPUTING ENVIRONMENTS WORKSHOP (GCE'08), 2008, Austin, Texas. **Proceedings...** Austin, Texas: IEEE, 2008, p. 1-10. Disponível em: <<https://arxiv.org/ftp/arxiv/papers/0901/0901.0131.pdf>>. Acesso em: 18 out. 2018.

GIL, Antonio Carlos. **Como Elaborar Projetos de Pesquisa**. 6. ed. São Paulo: Atlas, 2017. E-book. Disponível em: <<https://integrada.minhabiblioteca.com.br/books/9788597012934>>. Acesso em: 23 out. 2018.

GIL, Antonio Carlos. **Métodos e Técnicas de Pesquisa Social**. 6. ed. São Paulo: Atlas, 2008. 200 p. ISBN: 978-85-224-8261-0. E-book. Disponível em: <<https://integrada.minhabiblioteca.com.br/books/9788522484959>>. Acesso em: 23 out. 2018.

HERRMANN, Jiri et al. **Red Hat Enterprise Linux 7 Virtualization Deployment and Administration Guide**. [Raleigh]: Red Hat, 2018. E-book. Disponível em: <[https://access.redhat.com/documentation/en-us/red\\_hat\\_enterprise\\_linux/7/pdf/virtualization\\_deployment\\_and\\_administration\\_guide/Red\\_Hat\\_Enterprise\\_Linux-7-Virtualization\\_Deployment\\_and\\_Administration\\_Guide-en-US.pdf](https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/7/pdf/virtualization_deployment_and_administration_guide/Red_Hat_Enterprise_Linux-7-Virtualization_Deployment_and_Administration_Guide-en-US.pdf)>. Acesso em: 21 out. 2018.

HERRMANN, Jiri et al. **Red Hat Enterprise Linux 7 Virtualization Getting Started Guide**. [Raleigh]: Red Hat, 2018. E-book. Disponível em: <[https://access.redhat.com/documentation/en-us/red\\_hat\\_enterprise\\_linux/7/pdf/virtualization\\_getting\\_started\\_guide/Red\\_Hat\\_Enterprise\\_Linux-7-Virtualization\\_Getting\\_Started\\_Guide-en-US.pdf](https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/7/pdf/virtualization_getting_started_guide/Red_Hat_Enterprise_Linux-7-Virtualization_Getting_Started_Guide-en-US.pdf)>. Acesso em: 18 mai. 2019.

HILL, Mark Donald; JOUPPI, Norman Paul; SOHI, Gurindar. **Readings in computer architecture**. San Diego: Academic Press, 2000. 717 p. ISBN: 1-55860-539-8. E-book. Disponível em:

<<https://books.google.com.br/books?id=I7o8teBhz5wC&printsec=frontcover&hl=pt-BR#v=onepage&q&f=false>>. Acesso em: 03 out. 2018.

HUMPHREYS, Greg et al. Chromium: A Stream-Processing Framework for Interactive Rendering on Clusters. **ACM transactions on graphics (TOG)**, New York, New York, v. 21, n. 3, p. 693-702, 2002. Disponível em:

<<http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.375.136&rep=rep1&type=pdf>>. Acesso em: 10 out. 2018.

JACOB, Bart et al. **Introduction to Grid Computing**. [Armonk]: IBM Redbooks, 2005. E-book. Disponível em:

<<http://www.redbooks.ibm.com/redbooks/pdfs/sg246778.pdf>>. Acesso em: 18 out. 2018.

JAVED, Mobin; PAXSON, Vern. Detecting stealthy, distributed SSH brute-forcing. In: 2013 ACM SIGSAC Conference on Computer & Communications Security, 2013, Berlim, Alemanha. **Proceedings...** Berlim, Alemanha: ACM, 2013. p. 85-96.

Disponível em:

<<http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.439.1366&rep=rep1&type=pdf>>. Acesso em: 28 mai. 2019.

KATZ, J.; RIFER, W.; WILSON, A. R. EPEAT: Electronic Product Environmental Tool-development of an environmental rating system of electronic products for governmental/institutional procurement. In: 2005 IEEE INTERNATIONAL SYMPOSIUM ON ELECTRONICS AND THE ENVIRONMENT, 2005, New Orleans, Luisiana. **Proceedings...** New Orleans, Luisiana: IEEE, 2005, p. 1-6. Disponível em: <<https://www.computer.org/csdl/proceedings/isee/2005/8910/00/01436980.pdf>>. Acesso em: 22 out. 2018.

KHAN, Faisal H. et al. Challenges and Solutions in Measuring Computer Power Supply Efficiency for 80 Plus® Certification. In: 24<sup>th</sup> ANNUAL IEEE APPLIED POWER ELECTRONICS CONFERENCE AND EXPOSITION, 2009, Washington, DC. **Proceedings...** Washington, DC: IEEE, 2009, p. 2079-2085. Disponível em: <[http://www.ece.utah.edu/pearl/pdf/cp\\_2009\\_1.pdf](http://www.ece.utah.edu/pearl/pdf/cp_2009_1.pdf)>. Acesso em: 21 out. 2018.

KIVITY, Avi et al. kvm: the Linux Virtual Machine Monitor. In: LINUX SYMPOSIUM, 2007, Ottawa, Canadá. **Proceedings...** Ottawa, Canadá: [s. n.], 2007, p. 225-230. Disponível em: <<https://ols.fedoraproject.org/OLS/Reprints-2007/OLS2007-Proceedings-V1.pdf#page=225>>. Acesso em: 19 out. 2018.

KOPP, Ernani Maieski. **Construção de um cluster HPC para simulações de CFD**. 2012. 59f. Monografia de fim de curso (Especialização em Teleinformática e Redes de Computadores), Departamento Acadêmico de Eletrônica, Universidade Tecnológica Federal do Paraná, Paraná, Curitiba, 2012. Disponível em: <[http://repositorio.roca.utfpr.edu.br/jspui/bitstream/1/802/1/CT\\_TELEINFO\\_XX\\_2012\\_04.pdf](http://repositorio.roca.utfpr.edu.br/jspui/bitstream/1/802/1/CT_TELEINFO_XX_2012_04.pdf)>. Acesso em: 20 set. 2018.

KUDRYAVTSEV, Alexander; KOSHELEV, Vladimir; AVETISYAN, Arutvun. Modern HPC cluster virtualization using KVM and palacios. In: 19TH INTERNATIONAL CONFERENCE ON HIGH PERFORMANCE COMPUTING, 2012, Pune, Índia. **Proceedings...** Pune, Índia: IEEE, 2012. p. 1-9. Disponível em: <<https://www.computer.org/csdl/proceedings/hipc/2012/2372/00/06507495.pdf>>. Acesso em: 11 out. 2018.

KUROSE, Jim F.; ROSS, Keith W. **Redes de computadores e a internet**: uma abordagem top-down. 6. ed. São Paulo: Pearson Education do Brasil, 2013. 660 p. ISBN: 978-85-8143-677-7. E-book. Disponível em: <<http://univates.bv3.digitalpages.com.br/users/publications/9788581436777/>>. Acesso em: 07 out. 2018.

LEVINE, Steven. **Red Hat Enterprise Linux 7 High Availability Add-On Overview**. [Raleigh]: Red Hat, 2018. E-book. Disponível em: <[https://access.redhat.com/documentation/en-us/red\\_hat\\_enterprise\\_linux/7/pdf/high\\_availability\\_add-on\\_overview/Red\\_Hat\\_Enterprise\\_Linux-7-High\\_Availability\\_Add-On\\_Overview-en-US.pdf](https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/7/pdf/high_availability_add-on_overview/Red_Hat_Enterprise_Linux-7-High_Availability_Add-On_Overview-en-US.pdf)>. Acesso em: 17 out. 2018.

LIBVIRT. **Virtio - Libvirt Wiki**. Disponível em: <<https://wiki.libvirt.org/page/Virtio>>. Acesso em: 01 jun. 2019.

MARCONI, Marina Andrade, LAKATOS, Eva Maria. **Fundamentos de Metodologia Científica**. 8. ed. São Paulo: Atlas, 2017. E-book. Disponível em: <<https://integrada.minhabiblioteca.com.br/books/9788597010770>>. Acesso em: 23 out. 2018.

MARIN, Paulo Sérgio. **Data centers**: desvendando cada passo: conceitos, projeto, infraestrutura física e eficiência energética. São Paulo: Érica, 2013. ISBN: 978-85-365-0366-0.

MASCARENHAS, Sidnei Augusto. **Metodologia Científica**. São Paulo: Pearson Education do Brasil, 2012. ISBN: 978-85-64574-59-5.

MCGARVEY, Brian et al. Beowulf Cluster Design for Scientific PDE Models. In: 5<sup>TH</sup> ANNUAL LINUX SHOWCASE & CONFERENCE, 2001, Oakland, Califórnia. **Proceedings...** Berkeley, Califórnia: USENIX, 2001, 6 p. Disponível em: <[https://www.usenix.org/legacy/publications/library/proceedings/als01/full\\_papers/mcgarvey/mcgarvey.pdf](https://www.usenix.org/legacy/publications/library/proceedings/als01/full_papers/mcgarvey/mcgarvey.pdf)>. Acesso em: 13 out. 2018.

MOORE, Gordon E. Cramming more components onto integrated circuits. **Electronics**, [S. l.], v. 38, n. 8, abr. 1965. Disponível em: <<http://storage.jakstik.ac.id/intel-research/silicon/moorespaper.pdf>>. Acesso em: 01 out. 2018.

MURUGESAN, San. Harnessing Green IT: Principles and Practices. **IT Professional**, [S. l.], v. 10, n. 1, p. 24-33, 2008. Disponível em: <<http://www.pitt.edu/~dtipper/2011/GreenPaper.pdf>>. Acesso em: 21 out. 2018.

NEGUS, Christopher. **Fedora 10 and Red Hat Enterprise Linux Bible**. 1 ed. Indianapolis: Wiley, 2009. 1128 p. ISBN: 978-0-470-41339-5. E-book. Disponível em: <<https://www.oreilly.com/library/view/fedora-10-and/9780470413395/>>. Acesso em: 06 jun. 2019.

PEREIRA FILHO, Nelio Alves. **Serviços de pertinência para clusters de alta disponibilidade**. 2004. Dissertação (Mestrado em Ciência da Computação) - Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo, 2004. Disponível em: <<http://www.teses.usp.br/teses/disponiveis/45/45134/tde-04102004-104700/publico/dissertacao.pdf>>. Acesso em: 03 out. 2018.

PEREIRA JÚNIOR, João Eurípedes. **Especificação de serviço e suposições sobre o ambiente para um protocolo de alta disponibilidade**. 2010. Dissertação (Mestrado em Ciência da Computação) – Faculdade de Computação, Universidade Federal de Uberlândia, Minas Gerais, 2010. Disponível em: <<https://repositorio.ufu.br/bitstream/123456789/12506/1/Diss%20Joao.pdf>>. Acesso em: 06 jul. 2019.

PINHEIRO, Eduardo et al. Load balancing and unbalancing for power and performance in cluster-based systems. In: 2<sup>nd</sup> WORKSHOP ON COMPILERS AND OPERATING SYSTEMS FOR LOW POWER, 2001, Barcelona, Espanha. **Proceedings...** [Barcelona, Espanha]: [s. n.], 2001, p. 182-195. Disponível em: <[https://www.researchgate.net/profile/Enrique\\_Carrera/publication/2490475\\_Load\\_Balancing\\_and\\_Unbalancing\\_for\\_Power\\_and\\_Performance\\_in\\_Cluster-Based\\_Systems/links/00b4953445e20b6f0f000000.pdf](https://www.researchgate.net/profile/Enrique_Carrera/publication/2490475_Load_Balancing_and_Unbalancing_for_Power_and_Performance_in_Cluster-Based_Systems/links/00b4953445e20b6f0f000000.pdf)>. Acesso em: 15 out. 2018.

PLAŻEK, Joanna; BANAŚ, Krzysztof; KITOWSKI, Jacek. Comparison of message-passing and shared memory implementations of the GMRES method on MIMD computers. **Scientific Programming**, [S. l.], v. 9, n. 4, p. 195-209, 2001. Disponível em: <<http://downloads.hindawi.com/journals/sp/2001/681621.pdf>>. Acesso em: 29 set. 2018.

PRODANOV, Cleber Cristiano; FREITAS, Ernani Cesar. **Metodologia do Trabalho Científico: Métodos e Técnicas da Pesquisa e do Trabalho Acadêmico**. 2. ed. Novo Hamburgo: Feevale, 2013. 276 p. ISBN: 978-85-7717-158-3. E-book. Disponível em: <<http://www.feevale.br/Comum/midias/8807f05a-14d0-4d5b-b1ad-1538f3aef538/E-book%20Metodologia%20do%20Trabalho%20Cientifico.pdf>>. Acesso em: 23 out. 2018.

REIS, Adrieli Cristiane de Freitas et al. **Cluster de Alta Disponibilidade**. Guaratinguetá: Faculdade de Tecnologia de Guaratinguetá (FATEC-GT), [2011?]. Disponível em: <<http://www.academia.edu/download/31838709/cluster.pdf>>. Acesso em: 08 jul. 2019.

ROSTIROLLA, Gustavo. **Análise de desempenho e consumo energético de um cluster baseado em computadores de placa única**. 2014. 92p. Monografia de fim de curso (Bacharel em Engenharia da Computação), Centro Universitário Univates, Rio Grande do Sul, Lajeado, 2014.

RUTH, Stephen. Green IT — More Than a Three Percent Solution?. **IEEE Internet Computing**, [S. l.], n. 4, p. 74-78, 2009. – Disponível em: <<http://users.jyu.fi/~mieijala/Tietohallinnon%20johtaminen/Green%20IT/more%20than%20three%20percent%20solution.pdf>>. Acesso em: 21 out. 2018.

SCHEPKE, Claudio et al. Panorama de ferramentas para gerenciamento de clusters. In: III WORKSHOP DE PROCESSAMENTO PARALELO E DISTRIBUÍDO, 2005, Porto Alegre. **Proceedings...** Porto Alegre: UFRGS, 2005, 4 p. Disponível em: <[https://www.researchgate.net/profile/Marcelo\\_Neves2/publication/228630156\\_Panorama\\_de\\_ferramentas\\_para\\_gerenciamento\\_de\\_clusters/links/0fcfd50ec2a52ae1b8000000/Panorama-de-ferramentas-para-gerenciamento-de-clusters.pdf](https://www.researchgate.net/profile/Marcelo_Neves2/publication/228630156_Panorama_de_ferramentas_para_gerenciamento_de_clusters/links/0fcfd50ec2a52ae1b8000000/Panorama-de-ferramentas-para-gerenciamento-de-clusters.pdf)>. Acesso em: 17 out. 2018.

SCOGLAND, Tom; SUBRAMANIAM, Balaji; FENG, Wu-chun. The Green500 list: escapades to exascale. **Computer Science - Research and Development**, Blacksburg, v. 28, n. 2-3, p. 109-117, 2013. Disponível em: <<https://vtechworks.lib.vt.edu/bitstream/handle/10919/19474/paper.pdf?sequence=3>>. Acesso em: 10 out. 2018.

SLOAN, Joseph D. **High Performance Linux Clusters with OSCAR, Rocks, OpenMosix, and MPI: A Comprehensive Getting-Started Guide**. Sebastopol: O'Reilly Media, 2004. 370 p. ISBN: 0-596-00570-9. E-book. Disponível em: <<https://books.google.com.br/books?id=Xw8zLz3H8WAC&pg=PP1&hl=pt-BR&pg=PP1#v=onepage&q&f=false>>. Acesso em: 20 set. 2018.

SMITH, James E.; NAIR, Ravi. The Architecture of Virtual Machines. **Computer**, [S. l.], v. 38, n. 5, p. 32-38, 2005. Disponível em: <<https://eecs.ceas.uc.edu/~paw/classes/ece975/sp2010/papers/smith-05.pdf>>. Acesso em: 19 out. 2018.

STERLING, Thomas Lawrence; LUSK, Ewing. **Beowulf Cluster Computing with Windows**. Cambridge, Massachusetts: MIT Press, 2002. 445 p. ISBN: 0-262-69275-9. E-book. Disponível em: <<https://books.google.com.br/books?id=tyzGX00gcBoC&printsec=frontcover&hl=pt-BR#v=onepage&q&f=false>>. Acesso em: 07 out. 2018.

TANENBAUM, Andrew S.; STEEN, Maarten van. **Distributed Systems Principles and Paradigms**. New Jersey: Prentice-Hall, 2002. 785 p. ISBN: 0-13-088893-1. TOP500. **The Green500 List for June 2018**. Disponível em: <<https://www.top500.org/green500/lists/2018/06/>>. Acesso em: 21 out. 2018.

VALENTINI, Giorgio Luigi et al. An overview of energy efficiency techniques in cluster computing systems. **Cluster Computing**, [S. l.], v. 16, n. 1, p. 3-15, 2013. Disponível em: <<http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.402.2530&rep=rep1&type=pdf>>. Acesso em: 30 set. 2018.

VARRETTE, Sébastien et al. HPC Performance and Energy-Efficiency of Xen, KVM and VMware Hypervisors. In: 25<sup>th</sup> SYMPOSIUM ON COMPUTER ARCHITECTURE AND HIGH PERFORMANCE COMPUTING, 2013, Porto de Galinhas.

**Proceedings...** Porto de Galinhas: IEEE Computer Society Press, 2013, p. 89-96.

Disponível em:

<[https://www.researchgate.net/profile/Mateusz\\_Guzek/publication/258498122\\_HPC\\_performance\\_and\\_energy\\_efficiency\\_of\\_Xen\\_KVM\\_and\\_VMware\\_hypervisors/links/0046352f89b1fb445d000000.pdf](https://www.researchgate.net/profile/Mateusz_Guzek/publication/258498122_HPC_performance_and_energy_efficiency_of_Xen_KVM_and_VMware_hypervisors/links/0046352f89b1fb445d000000.pdf)>. Acesso em: 07 nov. 2018.

VERAS, Manoel. **Virtualização**: componente central do Datacenter. Rio de Janeiro: Brasport, 2011. 333 p. ISBN: 978-85-7452-462-2.

VOGELS, Werner et al. The Design and Architecture of the Microsoft Cluster Service: A Practical Approach to High-Availability and Scalability. In: TWENTY-EIGHTH ANNUAL INTERNATIONAL SYMPOSIUM ON FAULT-TOLERANT COMPUTING (FTCS'98), 1998, Munique, Alemanha. **Proceedings...** Munique, Alemanha: IEEE Computer Society Press, 1998, p. 422-431. Disponível em: <<https://arxiv.org/pdf/cs/9809006.pdf>>. Acesso em: 12 out. 2018.

YANG, Chao-Tung et al. Using a Beowulf cluster for a remote sensing application. In: 22ND ASIAN CONFERENCE ON REMOTE SENSING, 2001, Singapura.

**Proceedings...** Singapura, 2001, 6 p. Disponível em:

<<https://crisp.nus.edu.sg/~acrs2001/pdf/150CHANG.PDF>>. Acesso em: 10 out. 2018.

ZHU, Yanmin; NI, Lionel M. A Survey on Grid Scheduling Systems. **Department of Computer Science and Engineering, Shanghai Jiao Tong University**, Shanghai, China, v. 32, p. 1-47, 2013. Disponível em:

<[http://www.cs.sjtu.edu.cn/~yzhu/reports/SJTU\\_CS\\_TR\\_200309001.pdf](http://www.cs.sjtu.edu.cn/~yzhu/reports/SJTU_CS_TR_200309001.pdf)>. Acesso em: 18 out. 2018.



**UNIVATES**

R. Avelino Tallini, 171 | Bairro Universitário | Lajeado | RS | Brasil  
CEP 95900.000 | Cx. Postal 155 | Fone: (51) 3714.7000  
[www.univates.br](http://www.univates.br) | 0800 7 07 08 09