

Izaias de Faria

**UMA ABORDAGEM DE SELEÇÃO DE RECURSOS
CONSCIENTE DE CONSUMO DE ENERGIA BASEADA
EM TOPOLOGIA DE REDE, TAMANHO DE ARQUIVOS
E POTÊNCIA DE EQUIPAMENTOS**

Dissertação submetida ao Programa
de Pós-Graduação em Ciência da Com-
putação para a obtenção do Grau de
Mestre em Ciência da Computação.
Orientador: Prof. Dr. Mario Antonio
Ribeiro Dantas
Coorientador: Prof. Dr. Márcio Bas-
tos Castro

Florianópolis

2015

Ficha de identificação da obra elaborada pelo autor,
através do Programa de Geração Automática da Biblioteca Universitária da UFSC.

de Faria, Izaias

Uma Abordagem de Seleção de Recursos Consciente de Consumo de energia baseada em Topologia de Rede, Tamanho de Arquivos e Potência de Equipamentos / Izaias de Faria ; orientador, Mário Antonio Ribeiro Dantas ; coorientador, Márcio Bastos Castro. - Florianópolis, SC, 2015.
102 p.

Dissertação (mestrado) - Universidade Federal de Santa Catarina, Centro Tecnológico. Programa de Pós-Graduação em Ciência da Computação.

Inclui referências

1. Ciência da Computação. 2. Eficiência energética. 3. Seleção de recursos. 4. Nuvem computacional. I. Antonio Ribeiro Dantas, Mário. II. Bastos Castro, Márcio. III. Universidade Federal de Santa Catarina. Programa de Pós-Graduação em Ciência da Computação. IV. Título.

Izaias de Faria

**UMA ABORDAGEM DE SELEÇÃO DE RECURSOS
CONSCIENTE DE CONSUMO DE ENERGIA BASEADA
EM TOPOLOGIA DE REDE, TAMANHO DE ARQUIVOS
E POTÊNCIA DE EQUIPAMENTOS**

Esta Dissertação foi julgada aprovada para a obtenção do Título de “Mestre em Ciência da Computação”, e aprovada em sua forma final pelo Programa de Pós-Graduação em Ciência da Computação.

Florianópolis, 19 de fevereiro 2015.

Prof. Dr. Ronaldo dos Santos Mello
Coordenador do Curso

Banca Examinadora:

Prof. Dr. Mario Antonio Ribeiro Dantas
Orientador

Prof. Dr. Márcio Bastos Castro
Coorientador

Prof. Dr. Edson Borin
Universidade Estadual de Campinas

Prof(a). Dra. Carla Merkle Westphall
Universidade Federal de Santa Catarina

Prof(a). Dra. Patricia Della M ea Plentz
Universidade Federal de Santa Catarina

Este trabalho é dedicado aos meus colegas e principalmente a minha família, desistir é muito simples, mas decidir ir até o fim é uma jornada e uma decisão renovada diariamente, a minha pelo menos não foi trilhada sozinho.

AGRADECIMENTOS

Primeiramente gostaria de agradecer a minha família que sempre me incentivou na busca pelos estudos, desde a graduação e me ajuda a me manter focado em mais uma etapa da minha construção profissional. Em especial a minha irmã que foi quem me acompanhou de perto em minhas variações de humor e frustrações ao longo destes últimos dois anos.

Gostaria de agradecer ao professor Dr. Mario Dantas, que acreditou em minha capacidade como pesquisador e que ajudou a nutrir esta competência ao ponto de gerar este trabalho de pesquisa. Agradeço também ao professor Dr. Márcio Castro por aceitar coorientar este trabalho.

Com relação aos colegas de laboratório, gostaria de agradecer a minha amiga Eliza, pois foi a partir de sua defesa de mestrado que a ideia para esta dissertação surgiu, não apenas pela ideia do trabalho, como também pelas inúmeras conversas que tivemos ao longo dos últimos três anos.

Gostaria também de agradecer ao meu grande colega Eduardo Camilo pelas inúmeras discussões que tivemos durante uma convivência próxima de diária pelos últimos dois anos, que certamente contribuíram para a melhora deste trabalho.

De maneira geral gostaria de agradecer a todos os colegas que passaram pelo Laboratório de Pesquisa em Sistemas Distribuídos (LA-PESD) que de alguma maneira contribuíram direta ou indiretamente a construção desta obra, seja através de pequenas sugestões ou apenas discussões de ideias.

A Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) pela bolsa de estudos de Mestrado durante todo o período do mestrado.

Por fim, gostaria de agradecer imensamente a professora Miriam Capretz da Universidade de Western Ontario (UWO) por ter me acolhido durante o período em que estive em seu laboratório, e ao novo amigo Wilson Higashino e sua esposa Ana, tanto pelas contribuições ao trabalho quanto a minha recepção na Universidade de Western Ontario.

Uma pessoa inteligente aprende com os
seus erros, uma pessoa sábia aprende com
os erros dos outros.

(Augusto Cury)

RESUMO

Recentes avanços na área da Computação de Alto Desempenho (HPC) tem gerado uma grande variedade de possibilidades para pesquisas na área. Arquiteturas paralelas e distribuídas modernas apresentam um aumento considerável em sua capacidade de processamento. Entretanto, esse crescimento de desempenho é acompanhado por um aumento de consumo de energia. Neste cenário, a comunidade científica tem estudado técnicas voltadas à redução de consumo de energia em tais plataformas. Arquiteturas de alto desempenho são amplamente utilizadas em ambientes empresarial e acadêmico quando há a necessidade de grande poder computacional. Recentemente, infraestruturas legadas tem sido adaptadas ao modelo de nuvem computacional, o qual fornece recursos sob demanda e permite a usuários contratar serviços de infraestrutura, plataforma e *software*. Neste trabalho propomos uma abordagem genérica de alocação de recursos energeticamente eficiente que melhora a eficiência energética de ambientes de alto desempenho heterogêneos selecionando recursos menos custosos. A abordagem proposta considera o custo para transferência de dados, assim como o estado e eficiência energética dos nodos computacionais. Após realizados diversos experimentos em um ambiente simulado de nuvem, concluiu-se que, em alguns casos, a abordagem proposta reduz consideravelmente o consumo de energia em comparação com abordagens existentes na literatura.

Palavras-chave: Alto desempenho. Alocação de recursos. Eficiência energética. Nuvem computacional.

ABSTRACT

Recent advances in High Performance Computing (HPC) have led to a wide range of new possibilities for research. In this context, modern parallel and distributed architectures have presented a steady increase in their processing capabilities. However, such growth is usually followed by an increase in energy consumption. Because of that, the research community has been focusing on techniques to reduce energy consumption on such platforms. HPC architectures are now widely used in business and academic environments when high computing power is crucial. Recently, legacy structures have been adapted to the cloud computing model, which provides resources on demand such as infrastructure, software or platform. In this work we propose a generic energy-efficient scheduling approach that improves the energy efficiency of high performance heterogeneous environments by selecting the least costly resources. The proposed approach takes into consideration the cost of data transfers as well as the state and energy efficiency of computing nodes. After carrying out several experiments in a cloud simulated environment we concluded that, in some cases, the proposed approach achieves considerably better energy efficiency than other existing approaches in the literature.

Keywords: High performance computing. Energy consumption. Cloud computing.

LISTA DE FIGURAS

Figura 1	Taxonomia de Flynn	30
Figura 2	Exemplo de ambiente de <i>cluster</i> não dedicado e dedicado	33
Figura 3	Exemplo de ambiente <i>multi-cluster</i>	35
Figura 4	Arquitetura Genérica de um sistema de grade (KON; GOLDMAN, 2008)	37
Figura 5	Modelo de níveis da nuvem computacional	39
Figura 6	A arquitetura UMA (RIBEIRO et al., 2009)	41
Figura 7	A arquitetura NUMA (RIBEIRO et al., 2009)	42
Figura 8	A arquitetura NUMA atual (RIBEIRO et al., 2009)	43
Figura 9	Visão simplificada de um MPPA-256 (CASTRO et al., 2014)	44
Figura 10	Comparativo de Simuladores de Grade (FRANCO, 2011)	50
Figura 11	Interação entre RMS e recursos	64
Figura 12	Seleção de recurso em uma nuvem	65
Figura 13	Seleção de recurso em uma nuvem com abordagem ENA	66
Figura 14	Arquitetura proposta em libts	68
Figura 15	Topologia da rede	72
Figura 16	Recursos alocados por HTCondor	73
Figura 17	Recursos alocados por ENA	74
Figura 18	Comparação entre HTCondor e ENA	75
Figura 19	Topologia da rede UFSC	76
Figura 20	Cenário 1 10MB	78
Figura 21	Cenário 1 100MB	78
Figura 22	Cenário 1 1GB	79
Figura 23	Cenário 1 10GB	79
Figura 24	Cenário 1 100GB	79
Figura 25	Diferença percentual de energia consumida entre ENA e demais algoritmos para Cenário 1	82
Figura 26	Cenário 2 10MB	83
Figura 27	Cenário 2 100MB	83
Figura 28	Cenário 2 1GB	83
Figura 29	Cenário 2 10GB	84

Figura 30 Cenário 2 100GB	84
Figura 31 Diferença percentual de energia consumida entre ENA e demais algoritmos para Cenário 2	86

LISTA DE TABELAS

Tabela 1	Resumo comparativo entre os trabalhos relacionados...	62
Tabela 2	Descrição de equipamentos experimentos preliminares .	72
Tabela 3	Consumo de energia dos recursos alocados pelo sistema HTCondor	73
Tabela 4	Consumo de recursos alocados por ENA	74
Tabela 5	Descrição de equipamentos	76
Tabela 6	Configuração das máquinas simuladas	77
Tabela 7	Resultado completo Cenário 1 10MB	80
Tabela 8	Resultado completo Cenário 1 100MB	80
Tabela 9	Resultado completo Cenário 1 1GB	81
Tabela 10	Resultado completo Cenário 1 10GB	81
Tabela 11	Resultado completo Cenário 1 100GB	81
Tabela 12	Resultado completo Cenário 2 10MB	85
Tabela 13	Resultado completo Cenário 2 100MB	85
Tabela 14	Resultado completo Cenário 2 1GB	85
Tabela 15	Resultado completo Cenário 2 10GB	86
Tabela 16	Resultado completo Cenário 2 100GB	86

LISTA DE ABREVIATURAS E SIGLAS

CPU	Central Processing Unit	23
SLA	Service Level Agreement	23
VM	Virtual Machine	23
PM	Physical Machine	23
SMP	Symmetric Multiprocessors	33
SSI	Single System Image	34
SINAPAD	Sistema Nacional de Processamento de Alto Desempenho	36
NGS	National Grid Service	36
OV	Virtual Organization	36
NIST	National Institute of Standards and Technology	38
UMA	Uniform Memory Access	41
NUMA	Non-Uniform Memory Access	42
HPC	High Performance Computing	43
PE	Processing Elements	44
RM	Resource Manager	44
VLIW	Very long instruction word	44
NoC	Network-on-Chip	44
FIFO	First In First Out	45
LIFO	Last In First Out	45
SJF	Shortest Job First	45
SRT	Shortest Remaining Time Next	46
RR	Round-Robin	46
EDF	Earliest Deadline First	46
BFF	Backfill First Fit	46
BBF	Backfill Best Fit	46
WQ	Workqueue	46
WQR	Workqueue with Replication	47
Suff	Sufferage	47
XSuff	XSufferage	47
DFPLTF	Dynamic Fastest Processor to Largest Task First	48
FLOPS	Floating-point Operation Per Second	50
LIBTS	Library Tasks Scheduling	51

DVFS	Dynamic Voltage and Frequency Scaling	53
BoT	Bag-of-Tasks	55
EA	Energy-Aware	55
MRS	Most Recently Sleeping	55
QoS	Quality of Service	60
DVFS	Dynamic Voltage and Frequency Scaling	60
ENA	Energy and Network Aware	63
SDN	Software Defined Networks	90

SUMÁRIO

1	INTRODUÇÃO	23
1.1	CONTEXTUALIZAÇÃO	23
1.2	PERGUNTA DE PESQUISA	26
1.3	OBJETIVOS	26
1.3.1	Objetivo Geral	26
1.3.2	Objetivos Específicos	26
1.4	MÉTODO DE PESQUISA	27
1.5	ESCOPO DA DISSERTAÇÃO	27
1.6	ORGANIZAÇÃO DA DISSERTAÇÃO	28
2	SISTEMAS DISTRIBUÍDOS	29
2.1	INTRODUÇÃO	29
2.2	ARQUITETURAS COMPUTACIONAIS	30
2.2.1	Cluster	31
2.2.2	Multi-Cluster	34
2.2.3	Grades Computacionais	35
2.2.4	Nuvens Computacionais	38
2.2.5	UMA e NUMA	41
2.2.6	Manycoros	43
2.3	ESCALONAMENTO DE TAREFAS	45
2.4	SIMULAÇÃO	49
2.5	CONSIDERAÇÕES SOBRE O CAPÍTULO	51
3	TRABALHOS RELACIONADOS	53
3.1	INTRODUÇÃO	53
3.2	CONTROLE DE RECURSOS OCIOSOS	54
3.2.1	Ponciano e Brasileiro	54
3.2.2	Mämmelä et al.	55
3.2.3	Montes et al.	56
3.2.4	Laszewski et al.	57
3.3	ESCALONAMENTO VOLTADO A EFICIÊNCIA ENERGÉTICA	57
3.3.1	Montero et al.	57
3.3.2	Nesmachnow et al.	59
3.3.3	Teodoro et al.	59
3.3.4	Kim et al.	60
3.3.5	Garg e Buyya	60
3.3.6	Castro et al.	61
3.4	CONSIDERAÇÕES SOBRE O CAPÍTULO	61

4	ABORDAGEM ENA (ENERGY AND NETWORK AWARE)	63
4.1	INTRODUÇÃO	63
4.2	ARQUITETURA DA PROPOSTA	65
4.3	MÉTRICAS	67
4.4	MODELO DE EFICIÊNCIA ENERGÉTICA (MEE)	68
4.5	ALGORITMO ENA	69
5	AMBIENTE E RESULTADOS EXPERIMENTAIS	71
5.1	EXPERIMENTOS PRELIMINARES	71
5.2	AMBIENTE DE SIMULAÇÃO	75
5.3	RESULTADOS EXPERIMENTAIS	77
5.3.1	Cenário 1	78
5.3.2	Cenário 2	82
6	CONCLUSÕES E TRABALHOS FUTUROS	89
	REFERÊNCIAS	91
	APÊNDICE A – Publicações	101

1 INTRODUÇÃO

Neste capítulo são apresentados a contextualização e o tema de pesquisa. Na seção de pergunta de pesquisa é apresentada a hipótese para resolução do problema, bem como sua justificativa. A partir da pergunta de pesquisa e da hipótese são determinados os objetivos desta pesquisa. Por fim, são apresentados os métodos utilizados para a realização desta dissertação bem como a definição de seu escopo de pesquisa.

1.1 CONTEXTUALIZAÇÃO

A computação em nuvem oferece aos usuários serviços sob demanda, flexíveis, confiáveis e de baixo custo, cuja infraestrutura é denominada *datacenter* (ARMBRUST et al., 2009). Provedores de nuvem precisam construir e gerenciar estes *datacenters* com baixo custo, porém, com o aumento da escala de computação em nuvem observa-se também um crescimento no consumo de energia e nos custos operacionais. Um relatório relacionado a Microsoft (GREENBERG et al., 2008) mostra que os recursos físicos em um *datacenter* (por exemplo, CPU, memória, armazenamento, etc) são responsáveis por 45% do custo total e o custo com energia é de 15%, de acordo com Koomey (2007). Tendo em vista este cenário, os *datacenters* têm dado grande importância à redução no consumo de energia.

Hoje em dia, a maioria dos servidores físicos em um *datacenter* de uma nuvem utilizam tecnologia de virtualização. Baseado no *Service Level Agreement* (SLA) com os provedores da nuvem, os usuários solicitam um conjunto de *Virtual Machines* (VMs) que são alocadas em diferentes *hosts* e realizam comunicação entre si. Cada VM requer uma quantidade de recursos, como CPU, memória, armazenamento, largura de banda a fim de manter o desempenho da aplicação, isolamento e segurança. Além disso, a tecnologia de virtualização executa vários servidores virtuais na mesma *Physical Machine* (PM), o que é útil para melhorar a utilização de recursos, bem como, para a redução do consumo de energia.

A virtualização também pode ajudar os gestores da nuvem na implantação de recursos ordenada e sob demanda, o que proporciona uma solução eficaz para a gestão de recursos flexíveis e de baixo consumo de energia. Para nuvens públicas com virtualização, um dos seus

principais serviços é fornecer *Infraestrutura como Serviço* (IaaS), como o Amazon EC2 (EC2, 2014). Usuários pagam para alugar VMs com base em SLA, e provedores de nuvem tiram proveito da alocação flexível de VMs em PMs para otimizar a alocação de recursos de forma a atender às demandas dos usuários. Uma vez que a diferente utilização dos recursos é causada por diferentes mapeamentos entre VMs e PMs, a maior preocupação dos provedores de nuvens computacionais é a forma de se alocar várias VMs exigidas pelos usuários em servidores físicos de forma eficiente, de modo a minimizar o número de recursos ativos físicos e o consumo de energia, e, conseqüentemente, os custos de operação e de gestão.

A proliferação rápida dos serviços de computação em nuvem tem resultado em *datacenters* maciços, de grande porte. Provedores de serviços em nuvem consomem muitos megawatts de potência para operar esses *datacenters*, como Google (mais de 1.120GWh) e Microsoft (mais de 600GWh) (QURESHI, 2010). No entanto, nesses *datacenters* uma grande quantidade de energia excedente é provisionada, a fim de acomodar cargas de trabalho flutuantes e demandas de pico. Os servidores em *datacenters* geralmente operam entre 10% e 50% dos seus níveis máximos de utilização na maioria de seu tempo de execução (GUENTER; JAIN; WILLIAMS, 2011).

A consolidação de cargas de trabalho dinâmicas entre diferentes servidores baseados em tecnologias de virtualização (HERMENIER et al., 2009) tem sido extensivamente estudada a fim de melhorar a utilização dos recursos e reduzir o consumo de energia em *datacenters*. Especificamente, todas as máquinas virtuais (VMs) hospedando várias aplicações deverão ser consolidadas em um subconjunto de PMs via o processo de *VM migration* (LIU et al., 2013), enquanto outros PMs inativos (servidores) podem ser desligados para reduzir o consumo de energia. No entanto, a consolidação não é um processo tão trivial quanto conceitualmente alocar o número máximo de VMs para o número mínimo de PMs. Uma série de questões práticas conhecidas foram abordadas na literatura, como custo de migração de VMs (HERMENIER et al., 2009) (LIU et al., 2013) (VERMA; AHUJA; NEOGI, 2008), contenção de recursos e interferência de desempenho entre VMs co-localizadas (ZHU; ZHU; AGRAWAL, 2010).

Economias significativas no orçamento de energia de um *datacenter*, sem sacrificar SLAs, são um excelente incentivo econômico para os operadores de centros de dados, além de simbolizar uma contribuição significativa para uma maior sustentabilidade ambiental. De acordo com as estimativas da Amazon.com (HAMILTON, 2009), as despesas re-

lacionadas com o custo e operação dos servidores em *datacenters* são responsáveis por 53% do orçamento total (com base em um cronograma de amortização de 3 anos), enquanto custos relacionados com a energia equivalem a 42% do total, e incluem tanto o consumo de energia direta (19%) quanto a refrigeração da infra-estrutura (23%), amortizados por um período de 15 anos.

Dennis Pamlin, o supervisor global de Políticas do World Wildlife Fund(WWF)(PAMLIN, 2008) destaca diferentes soluções de TI e seu impacto benéfico sobre gases estufa, que incluem emissões de CO₂. Estas oportunidades incluem soluções baseadas em TI: por exemplo, edifícios, transporte e comunicação inteligente, bem como comércio, serviços e produção industrial inteligente. O termo coloquial “inteligente”, neste caso, significa “com baixa produção de carbono”, mostrando que a adoção de tais soluções de TI “inteligentes” permitirá um potencial grande de redução dos gases estufa, incluindo *Tecnologias da Informação e Comunicação* (TIC) em si, que são um grande consumidor de energia (e, portanto, um emissor de gases de efeito estufa), e soluções de TI que têm um enorme potencial de impacto na redução das emissões de gases estufa em muitos setores.

Economias de energia da ordem de 20% podem ser atingidas em consumo de energia de servidor e rede em relação aos níveis atuais (SiliconValley Leadership Group, 2008), e essas economias podem induzir a uma redução adicional de 30% realacionadas a necessidades de resfriamento, como detalhado em um estudo realizado pela HP e *Uptime Institute* (MALONE; BELADY, 2006). Este estudo mostra que a maior parte da energia de *datacenters* é gasta em equipamentos de refrigeração de TIC (entre 60% e 70%). Assim, existem inúmeros ganhos ambientais a serem obtidos a partir de uma pesquisa séria sobre eficiência energética na área geral de TI e redes de computadores.

A computação em nuvem é uma técnica de virtualização inerentemente eficiente em termos energéticos (HEWITT, 2008), em que os serviços são executados remotamente numa nuvem de computação ubíqua que fornece recursos escaláveis e virtualizados. Assim, os picos de carga podem ser transferidos para outras partes da nuvem e a agregação de recursos de uma nuvem pode proporcionar melhor utilização do *hardware*. Segundo Monteiro, Dantas e Rodriguez (2014) é possível se atingir uma economia de consumo de energia em um data-center em cerca de 37% utilizando-se gerenciamento de recursos voltado a eficiência energética.

Diante desta realidade, este trabalho de pesquisa visa propor uma abordagem de seleção de recursos que reduza o consumo de energia

de um ambiente distribuído. A fim de validar a abordagem proposta foi desenvolvido um algoritmo capaz de realizar estimativas de consumo de energia relacionadas ao ambiente.

1.2 PERGUNTA DE PESQUISA

Este trabalho de pesquisa busca responder a seguinte pergunta: **Como reduzir o consumo de energia de um ambiente de alto desempenho distribuído?** Para responder a essa pergunta é sugerida uma abordagem de seleção de recursos consciente do consumo de energia. Neste trabalho é considerado o ambiente de alto desempenho heterogêneo de uma nuvem computacional, buscando-se obter uma redução do consumo de energia sem impactar drasticamente a escalabilidade e desempenho geral do sistema. A abordagem proposta considera a possibilidade de variação de estado das máquinas presentes no ambiente observado. A fim de estimar o consumo geral da abordagem são realizadas estimativas que abrangem: 1) consumo de transferência de dados, diretamente relacionada ao tipo de equipamento utilizado no ambiente; 2) consumo de execução de tarefa em cada unidade de execução, seja a unidade um servidor, cluster ou grade; e por fim 3) consumo de variação de estado da unidade executora.

1.3 OBJETIVOS

De acordo com a pergunta de pesquisa, esta pesquisa apresenta os objetivos geral e específicos descritos a seguir.

1.3.1 Objetivo Geral

O objetivo geral desta dissertação é investigar uma abordagem que possibilite a redução do consumo de energia de ambientes de alto desempenho heterogêneos.

1.3.2 Objetivos Específicos

Dentre os principais objetivos específicos pode-se citar:

- Identificar um modelo para cálculo de consumo energético em

ambientes de alto desempenho heterogêneos;

- Propor uma abordagem para seleção de recursos consciente de energia para ambientes de alto desempenho heterogêneos;
- Selecionar um tipo de ambiente de alto desempenho para validar a abordagem de seleção proposta;
- Validar a abordagem com relação a estratégias conhecidas mais utilizadas no tipo de ambiente selecionado.

1.4 MÉTODO DE PESQUISA

As etapas realizadas para alcançar os objetivos desta dissertação foram:

- Pesquisa do estado da arte na área de consumo de energia em grades e nuvens computacionais;
- Estudo da arquitetura e ambiente simulado selecionado;
- Implementação do algoritmo de seleção de recursos baseado no modelo proposto;
- Aplicação do algoritmo em um ambiente de nuvem computacional simulado;
- Análise dos resultados gerados a partir dos estudos de caso.

1.5 ESCOPO DA DISSERTAÇÃO

Considerando os objetivos definidos, este trabalho propõe uma abordagem de seleção de recursos genérica, que pode ser aplicada a ambientes de alto desempenho heterogêneos. Ao longo do trabalho são discutidas possíveis arquiteturas nas quais a abordagem proposta pode ser aplicada. Das arquiteturas mencionadas, a abordagem é aplicada a um ambiente de nuvem computacional. O trabalho descreve também um modelo para o cálculo referente ao consumo energético e em seguida define um algoritmo que aplica o modelo. O algoritmo é testado em um ambiente simulado e comparado a outros algoritmos identificados na revisão bibliográfica.

1.6 ORGANIZAÇÃO DA DISSERTAÇÃO

Esta dissertação é dividida em 6 capítulos. No *Capítulo 2 - Sistemas Distribuídos* são apresentados os principais conceitos sobre sistemas distribuídos. Além disso, são apresentados os diferentes tipos de arquitetura e *software* utilizados nesses ambientes. Por fim, é tratado o tema de escalonamento de tarefas nesses ambientes.

O *Capítulo 3 - Trabalhos Relacionados* são destacados alguns trabalhos relacionados à área desse trabalho.

O *Capítulo 4 - Abordagem ENA* apresenta o modelo e abordagem propostos, bem como os detalhes da implementação do modelo em forma de um algoritmo.

O *Capítulo 5 - Ambiente e Resultados Experimentais* descreve o ambiente utilizado para a obtenção dos resultados experimentais, bem como os estudos de casos realizados como prova do funcionamento do modelo de seleção proposto.

O *Capítulo 6 - Conclusões e Trabalhos Futuros* apresenta as conclusões sobre a pesquisa realizada, bem como as indicações para trabalhos futuros.

2 SISTEMAS DISTRIBUÍDOS

Este capítulo tem como objetivo o detalhamento da arquitetura e dos componentes de *hardware* e *software* necessários para a construção de uma infraestrutura distribuída. É também apresentado o processo de escalonamento de tarefas em recursos distribuídos, com foco maior para estratégias mais utilizadas em ambientes de grades computacionais. Por fim, o capítulo descreve vantagens de se utilizar simulações em testes com ambientes de alto desempenho, bem como vantagens referentes a adoção do simulador Simgrid neste trabalho.

2.1 INTRODUÇÃO

A partir da década de 90 notou-se uma disponibilização de conexões de rede mais rápidas, bem como um aumento na capacidade de recursos em computadores pessoais. A partir deste cenário, a interconexão de computadores, local ou geograficamente distribuídos, tornou-se uma forma de aumentar o poder computacional. Com isso deu-se a popularização de sistemas distribuídos.

Coulouris, Dollimore e Kindberg (2007), definem um sistema distribuído como sendo um sistema no qual os componentes de *hardware* ou *software*, localizados em computadores interconectados por uma rede, comunicam-se e coordenam-se através da troca de mensagens. De acordo com Tanenbaum e Steen (2002), um sistema distribuído é uma coleção de computadores independentes que se apresenta com uma imagem única de sistema aos seus usuários. Para esses autores, a construção de um ambiente distribuído deve atender aos seguintes requisitos:

- **Tornar os recursos acessíveis:** facilitar para o usuário o acesso a recursos remotos e possibilitar o compartilhamento de forma controlada e eficiente;
- **Transparência na distribuição:** esconder do usuário o fato de que os processos e recursos estão fisicamente distribuídos em vários computadores;
- **Abertura:** oferecer ao usuário serviços de acordo com as regras padrão que descrevem sua sintaxe e semântica;
- **Escalabilidade:** a fim de atender a esse requisito, o ambiente

distribuído deve adicionar facilmente usuários e recursos no sistema mesmo que estes estejam fisicamente distantes, como por exemplo em organizações administrativamente independentes.

2.2 ARQUITETURAS COMPUTACIONAIS

A classificação de arquitetura de computadores mais aceita é conhecida como taxonomia de Flynn, proposta por Michael J. Flynn em 1972 (FLYNN, 1972). A taxonomia leva em consideração a quantidade de instruções executadas em paralelo pelo conjunto de dados para os quais as instruções são submetidas. A Figura 1 apresenta as classificações de computadores, que são subdivididas em: SISD, SIMD, MISD e MIMD (FLYNN, 1972).

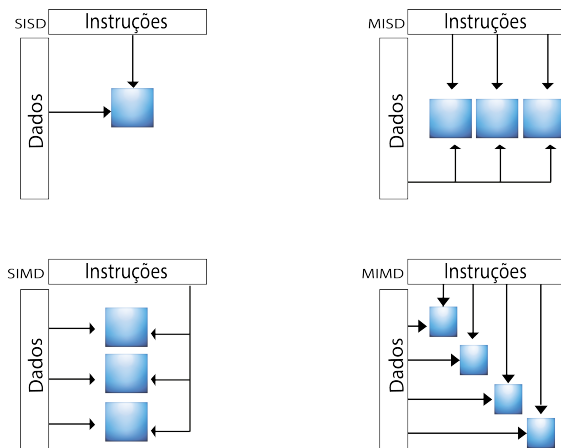


Figura 1: Taxonomia de Flynn

- **SISD** (*Single Instruction Single Data*): executa uma instrução de um programa por vez, ou seja, são computadores que apresentam apenas um processador. Este modelo pode ser representado por um computador pessoal mono-processado, onde as instruções são executadas sequencialmente.
- **SIMD** (*Single Instruction Multiple Data*): executa uma única instrução, porém esta instrução é processada sob diferentes itens de dados. Isso ocorre devido à existência de facilidades de *hardware*

para armazenamento e processamento de dados, como instruções vetoriais.

- **MISD** (*Multiple Instruction Single Data*): neste tipo de arquitetura são executadas múltiplas instruções sob um único conjunto de dados. No entanto, não se tem conhecimento da existência de computadores com essa classificação.
- **MIMD** (*Multiple Instruction Multiple Data*): executa múltiplas instruções sob múltiplos dados. Computadores com esta arquitetura possuem múltiplos processadores independentes, cada um podendo executar um conjunto diferente de instruções sob diferentes conjunto de dados.

As arquiteturas MIMD são classificadas, de modo geral, como multiprocessadores e multicomputadores, nas quais se diferenciam pelo compartilhamento ou não de memórias (MEFFE; MUSSI; MELLO, 2006).

- **Multiprocessadores**: arquitetura conhecida como fortemente acoplada, uma vez que processadores e memórias são interligados através de um sistema local de interconexão. Estas arquiteturas são caracterizadas por possuírem diversos processadores compartilhando uma ou um conjunto de memórias. A escalabilidade de uma arquitetura multiprocessada chega a milhares de processadores.
- **Multicomputadores**: nesta configuração cada processador possui suas próprias memórias locais. Diante disto, é conhecida como fracamente acoplada, uma vez que não há um compartilhamento forte, ou seja, a comunicação é efetuada através da troca de mensagens entre os processos em execução nos processadores.

2.2.1 Cluster

A ideia inicial de *cluster* computacional foi desenvolvida nos anos 60 pela IBM como uma forma de conectar grandes *mainframes* para prover um paralelismo comercial de baixo custo (BUYA, 1999).

Diante disso, os *clusters*, conhecidos também em português como agregados computacionais, são definidos por Buya (1999) como um sistema de processamento paralelo ou distribuído formado por uma coleção de computadores interconectados que trabalham juntos como um único recurso computacional integrado.

Por outro lado, Dantas (2005) entende que as configurações de *clusters* podem ser como uma agregação de computadores de forma dedicada, ou não, para a execução de aplicações específicas de uma organização.

Um *cluster* é composto por dois ou mais computadores (também chamados de nós) com um único processador ou por sistemas multiprocessados, interconectados por uma rede. Seu principal objetivo é realizar o processamento da aplicação de forma distribuída e transparente.

Segundo Buyya (1999) e Dantas (2005) os *clusters* podem ser classificados de acordo com limite geográfico, utilização dos nós, tipo de *hardware*, aplicação alvo, sistema operacional dos nós e tipos de nós.

- **Limite Geográfico:** baseado em sua localização e quantidade, os *clusters* podem ser classificados como: pequenos, constituídos em salas e laboratórios; médios, em nível de departamento; grande, para organizações.
- **Aplicação Alvo:** existem dois alvos principais às aplicações comuns: as aplicações que necessitam de um alto desempenho para a sua execução e as aplicações que precisam de alta disponibilidade. Aplicações que procuram o primeiro alvo se preocupam com o número de processadores, quantidade de memória e espaço em disco. Por outro lado, as que tem foco no segundo alvo são caracterizadas por não tolerarem interrupções. Entretanto, é possível observar que existem aplicações que exigem os dois tipos de requisitos.
- **Utilização dos Nós:** pode ser estabelecida através da participação não dedicada ou dedicada dos nós que irão compor o *cluster*. Em *clusters* não dedicados, as aplicações são executadas por meio da utilização dos ciclos de processador ociosos de máquinas pertencentes ao *cluster*. Por outro lado, os dedicados são projetados para executarem exclusivamente aplicações submetidas ao *cluster*. A Figura 2 apresenta um exemplo das configurações de *cluster* não dedicado e dedicado. Como pode ser visto, na configuração não dedicada convencional inúmeros computadores compartilham um único meio de comunicação, que é a rede local. Além disso, cada computador possui um conjunto de aplicativos locais e periféricos necessários para a execução das tarefas de seu usuário local. Na configuração dedicada pode-se observar que os nodos são interligados por um dispositivo de rede do tipo *switch* e nenhum computador dispõe de monitor, teclado ou *mouse*, mostrando assim sua dedicação à execução de aplicações.

- **Tipo de *Hardware*:** os tipos de *hardware* empregados nos *clusters* são classificados como: NOWs, CoPs ou PoPs, COWs e Clumps.

As NOWs (*Network of Workstations*) são caracterizadas pelo uso de estações de trabalho distribuídas numa rede local para compor um ambiente de *cluster*. De forma similar às NOWs, os CoPs (*Cluster of PCs*) ou PoPs (*Pile of PCs*) utilizam computadores pessoais do tipo PCs para formar o *cluster*.

No caso das COWs (*Cluster of Workstations*), estas são geralmente constituídas por máquinas dedicadas e homogêneas voltadas à execução de aplicações específicas. Além disso, estas dispõem de uma rede específica para interconexão das máquinas.

Os Clumps (*Cluster of SMPs*) são compostos de máquinas com arquiteturas SMP (*Symmetric Multiprocessors*), que usam memória compartilhada.

- **Tipos de Nós:** os *clusters* podem ser classificados quanto à similaridade de *software* e *hardware* dos nós que compõem o ambiente, ou seja, podem ser homogêneos ou heterogêneos.

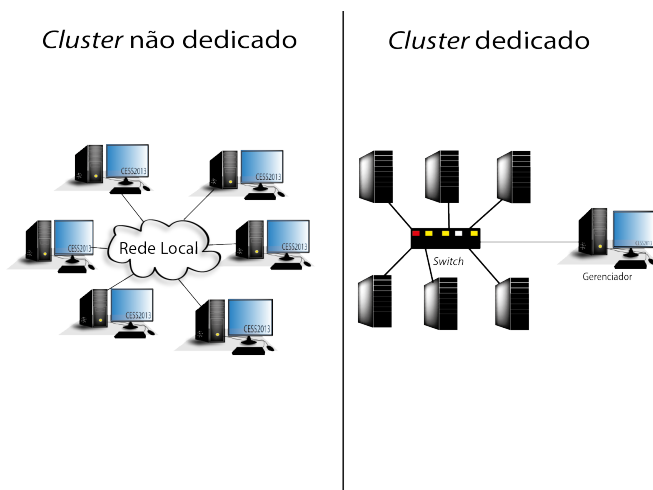


Figura 2: Exemplo de ambiente de *cluster* não dedicado e dedicado

Nos *clusters* homogêneos todos os nós possuem arquiteturas semelhantes e executam o mesmo sistema operacional. Por outro

lado, os heterogêneos possuem nós com arquiteturas diferentes e executam sistemas operacionais diferentes.

Contudo, muitas vezes, um único *cluster* não possui capacidade de resolver problemas computacionais que demandam grande poder computacional. Neste caso, múltiplos *clusters* interconectados, formam um ambiente *multi-cluster* e, conseqüentemente, podem aumentar o poder computacional para a resolução desses problemas. A seção seguinte apresenta uma descrição mais detalhada desses ambientes.

2.2.2 Multi-Cluster

Conforme visto na seção anterior, os *clusters* possuem a localização restrita a um único domínio. No entanto, é possível, e muitas vezes necessário, que *clusters* de múltiplos domínios se interconectem para formar um sistema distribuído único de larga escala. Estes sistemas são conhecidos como *multi-cluster* (FERREIRA, 2010).

Diferentemente dos *clusters*, que são formados por um conjunto independente de estações de trabalho interconectadas por uma rede local (LAN), os *multi-clusters* são formados por um conjunto de *clusters* interconectados por uma rede WAN (*World-Area Network*).

Os sistemas *multi-cluster* podem ser classificados como *super-cluster* e *cluster-de-cluster* (JAVADI; AKBARI; ABAWAJY, 2006). O primeiro pode ser caracterizado por possuir um grande número de processadores homogêneos e heterogeneidade nas redes de comunicação. Por outro lado, o *cluster-de-cluster* é construído pela interconexão de múltiplos *clusters*, porém com heterogeneidade tanto nas redes de comunicação quanto nos processadores.

Existe um alto grau de complexidade nos ambientes *multi-cluster* quanto aos problemas de escalonamento de tarefas, pois seus recursos são: heterogêneos, distribuídos e altamente compartilhados no tempo e no espaço. Diante disso, o recebimento contínuo de tarefas e as mudanças dinâmicas da disponibilidade da capacidade de processador dificultam a utilização de algoritmos tradicionalmente utilizados em sistemas de *cluster* (ABAWAJY; DANDAMUDI, 2003).

A configuração de sistemas *multi-cluster* forma um sistema de imagem única (SSI - *Single System Image*), ou seja, fornece ao usuário uma visão unificada do compartilhamento de recursos e serviços do ambiente (DANTAS, 2005).

A Figura 3 apresenta um exemplo de ambientes *multi-cluster*.

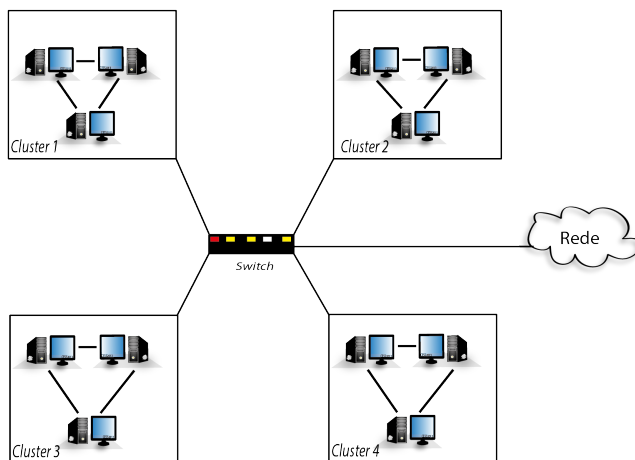


Figura 3: Exemplo de ambiente *multi-cluster*

2.2.3 Grades Computacionais

Na década de 90, com a maior disponibilidade de conexões mais rápidas e com aplicações necessitando cada vez mais de poder computacional, pesquisadores começaram a imaginar uma infraestrutura computacional que conectasse computadores e *clusters* geograficamente distribuídos para atender essa demanda. Diante disso, surgiu a proposta da chamada grade computacional, *Computational Grid* (FOSTER; KESSELMAN, 2011). Essa nomenclatura é uma analogia à rede elétrica (*power grid*) que disponibiliza a energia sem se preocupar em explicar onde esta foi gerada e como está sendo destinada. A ideia de uma grade computacional é criar um ambiente computacional distribuído que possua mecanismos que permitam o processamento, armazenamento e uso dos recursos de forma transparente para o usuário (DANTAS, 2005).

De acordo com Foster e Kesselman (2004), uma grade computacional é uma infraestrutura de *hardware* e *software* que provê acesso confiável, abrangente e barato a capacidades computacionais. Já para Dantas (2005), uma grade computacional pode ser entendida como uma plataforma heterogênea de computadores geograficamente distribuídos, onde usuários acessam seus recursos através de uma interface única.

Krauter, Buyya e Maheswaran (2002) propõem uma taxonomia para identificar os tipos de sistemas de grades existentes. Para tanto, os sistemas podem ser divididos nas seguintes categorias:

- **Grade Computacional** (*Computational Grid*): são sistemas que unem recursos geograficamente distribuídos para obter alta capacidade de processamento. Dependendo de como esta capacidade é utilizada, esse sistema pode ser subdividido em: supercomputação distribuída e grade de alta taxa de transferência.

Uma grade de supercomputação distribuída executa a aplicação paralelamente entre múltiplas máquinas para reduzir o tempo de conclusão da tarefa. Já uma grade de alta taxa de transferência aumenta a taxa de conclusão de um fluxo de tarefas.

- **Grade de Dado** (*Data Grid*): consiste de uma infraestrutura para acesso, pesquisa e processamento de informações a partir de repositórios de dados, que estão geograficamente distribuídos.
- **Grade de Serviço** (*Service Grid*): são sistemas que provêm serviços que não são oferecidos por máquinas simples. Normalmente esse tipo de infraestrutura é administrada por grandes instituições e reúne alto desempenho, recursos computacionais dedicados, como *clusters*, supercomputadores e grandes sistemas de armazenamento de dados (BRASILEIRO et al., 2008). Como exemplo desse tipo de grade pode-se citar: SINAPAD no Brasil (SINAPAD, 2014) e NGS no Reino Unido (NGS, 2014).

Uma configuração de grade é composta por organizações virtuais (*Virtual Organizations*), que podem ser indivíduos ou entidades que compartilham seus recursos, porém apresentam determinadas políticas quanto ao acesso e uso e algumas restrições quanto à disponibilidade. Diante disso, tem-se proposto arquiteturas que permitam a interoperabilidade entre diferentes organizações virtuais. Camargo et al. (2006) apresentam uma arquitetura genérica implementada pela maioria dos sistemas de grades. Essa arquitetura é ilustrada na Figura 4 e possui cinco serviços, que serão descritos a seguir.

- **Agente de Acesso:** é a primeira interface de acesso para os usuários, permitindo a interação entre usuário e grade. Está presente em cada nó que solicite a execução de aplicações.
- **Serviço Local de Oferta de Recursos:** executado em cada máquina que disponibiliza seus recursos para a grade. É res-

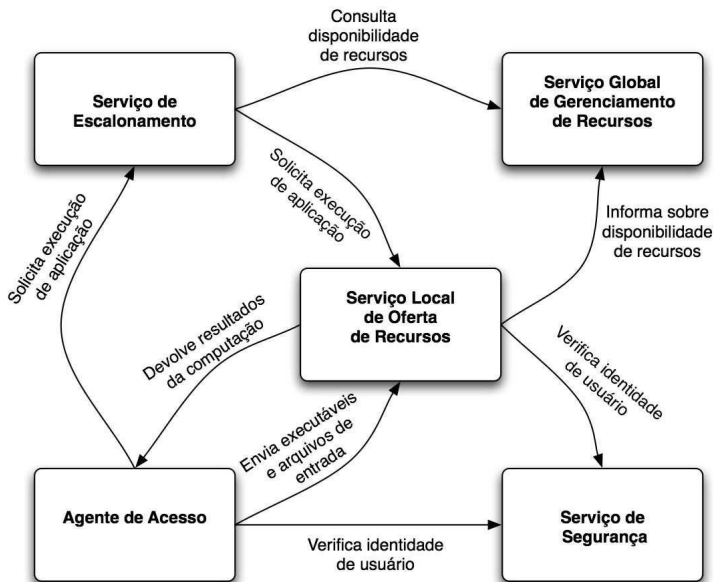


Figura 4: Arquitetura Genérica de um sistema de grade (KON; GOLDMAN, 2008)

ponsável por iniciar a execução da aplicação, reportar erros e retornar resultados.

- **Serviço Global de Gerenciamento de Recursos:** é responsável por monitorar o estado dos recursos compartilhados e por responder a solicitações de uso destes recursos, a fim de combinar as requisições com os recursos oferecidos.
- **Serviço de Escalonamento:** é responsável por escalonar as aplicações para recursos disponíveis. Para isso, recebe as solicitações de execução da aplicação, obtém informações quanto à disponibilidade dos recursos junto ao Serviço Global de Gerenciamento de Recursos e determina onde cada aplicação será executada.
- **Serviço de Segurança:** é responsável por proteger os recursos compartilhados para que o nó que compartilha seus recursos não sofra ataques de aplicações maliciosas. Além disso, é responsável

por autenticar os usuários de modo que se saiba quem é o responsável pelas aplicações submetidas, bem como a segurança das comunicações da grade, mantendo assim, a confiabilidade e integridade dos dados.

2.2.4 Nuvens Computacionais

Nuvem computacional, do inglês *Cloud Computing*, tem sido muito utilizada quando deseja-se obter recursos computacionais sob demanda. Apesar da ascensão de seu uso nos dias de hoje, a ideia básica de nuvens computacionais não é nova, tendo sido prevista nos anos 60 pelo cientista da computação John McCarthy, quando mencionou que a computação podia um dia ser organizada como de utilidade pública exatamente como o sistema de telefonia (GARFINKEL, 1999).

O conceito de nuvem computacional pode ter diferentes percepções, considerando que esta não é uma tecnologia nova e sim uma junção de tecnologias existentes executadas de forma diferenciada.

Segundo Foster et al. (2008) nuvem computacional é um paradigma de computação distribuída de larga escala que é impulsionado pela economia, na qual um conjunto de recursos abstratos, virtualizados, dinamicamente escaláveis, com poder computacional gerenciável, além de plataformas e serviços, são proporcionados para usuários externos através da Internet. Por outro lado, segundo a NIST (MELL; GRANCE, 2011), responsável por desenvolver padrões e diretrizes, nuvem computacional é um modelo para permitir acesso à rede sob demanda de forma ubíqua e conveniente para o compartilhamento de recursos computacionais configuráveis (por exemplo, rede, servidores, armazenamento, aplicações e serviços) que pode ser rapidamente fornecido e liberado com o mínimo de esforço de gerenciamento ou de interação com o provedor de serviço.

Existem muitas definições para o modelo de arquitetura de nuvem computacional, no entanto, de acordo com Zhang, Cheng e Boutaba (2010) a arquitetura está dividida em modelo de níveis, modelo de serviços e tipos de nuvens. A seguir são detalhadas cada uma das divisões.

- **Modelo de Níveis:** a arquitetura de nuvem computacional pode, de modo geral, ser subdividida em quatro níveis, apresentados na Figura 5: nível de *hardware*, nível de infraestrutura, nível de plataforma e nível de aplicação.

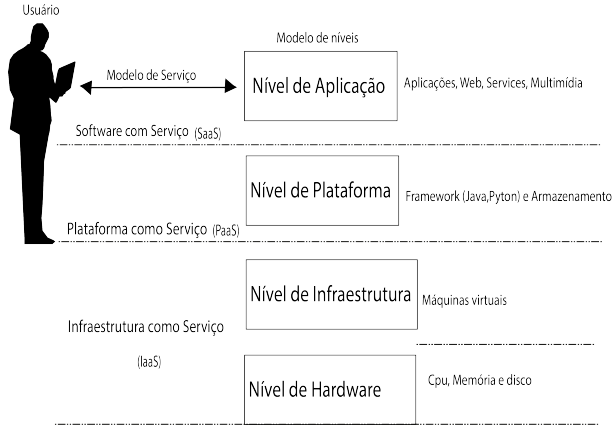


Figura 5: Modelo de níveis da nuvem computacional

- **Nível de *Hardware*:** este nível é responsável por gerenciar recursos físicos como: servidores, roteadores e *switches*.
- **Nível de *Infraestrutura*:** também conhecido como nível de virtualização, é um componente essencial da nuvem, pois cria um *pool* de armazenamento e recursos computacionais através do particionamento lógico de recursos físicos utilizando tecnologias de virtualização.
- **Nível de *Plataforma*:** adiciona uma coleção de ferramentas, *middleware* e serviços especializados para proporcionar uma plataforma de desenvolvimento (FOSTER et al., 2008).
- **Nível de *Aplicação*:** contém a aplicação que será executada na nuvem.

Cada nível da arquitetura é fracamente acoplado com os níveis acima e abaixo, permitindo que cada um se desenvolva separadamente. A arquitetura modular possibilita que a nuvem computacional suporte grandes quantidades de requisições das aplicações, reduzindo a sobrecarga de gerenciamento e manutenção.

- **Modelo de Serviços:** nesse modelo, todos os níveis da arquitetura da nuvem computacional podem ser implementados como um serviço para o nível acima. Consequentemente, todos os níveis podem ser percebidos como clientes do nível abaixo (ZHANG;

CHENG; BOUTABA, 2010). Entretanto, em geral, as nuvens computacionais oferecem serviços que podem ser agrupados em três categorias, conforme apresentados na Figura 5:

- **Infraestrutura como Serviço (IaaS)**: refere-se ao fornecimento de processamento, armazenamento, rede e outros serviços computacionais fundamentais capazes de desenvolver e executar *software*, incluindo sistemas operacionais e aplicativos. No entanto, o usuário não gerencia ou controla a infraestrutura da nuvem, mas possui controle sobre os sistemas operacionais, armazenamento e desenvolvimento de aplicações (MELL; GRANCE, 2011).
 - **Plataforma como Serviço (PaaS)**: oferece ao usuário um ambiente para desenvolvimento de aplicações personalizadas. Geralmente os desenvolvedores precisam respeitar algumas restrições como o tipo de *software* que eles podem utilizar. Nessa categoria, o usuário não gerencia ou controla a infraestrutura da nuvem, o que inclui rede, servidores, sistemas operacionais e armazenamento.
 - **Software como Serviço (SaaS)**: refere-se ao fornecimento de aplicações acessadas remotamente pelos usuários através da Internet.
- **Tipos de Nuvens**: de acordo com a NIST (MELL; GRANCE, 2011), existem quatro tipos de nuvens computacionais:
- **Nuvens Públicas**: uma nuvem na qual provedores de serviço oferecem seus recursos computacionais para o público em geral, baseado em um modelo sob demanda.
 - **Nuvens Privadas**: são projetadas para uso exclusivo de uma única organização, podendo ser construídas e gerenciadas pela organização ou por provedores externos.
 - **Nuvens de Comunidades**: a infraestrutura da nuvem é compartilhada por várias organizações e suporta uma comunidade específica que possui as mesmas preocupações, tais como: missão, requisições seguras e políticas.
 - **Nuvens Híbridas**: a infraestrutura da nuvem é composta por duas ou mais nuvens (pública, privada ou de comunidade) unidas por tecnologia padronizada ou proprietária que permite a portabilidade de dados e aplicações.

2.2.5 UMA e NUMA

Neste trabalho, é considerada como uma arquitetura hierárquica de memória compartilhada, qualquer plataforma com multiprocessador que possui unidades: (i) de processamento que compartilham uma memória global e (ii) unidades de processamento e componentes de memória organizada em alguma forma de topologia hierárquica.

Em plataformas UMA (*Uniform Memory Access*), todas as unidades de processamento possuem custos de acesso semelhantes a memória compartilhada global. Isto é devido ao fato de que a memória compartilhada global está ligada a um único barramento, que é utilizado pelas unidades de processamento para acessar a memória. Além disso, nesta arquitetura os elementos de processamento compartilham os dispositivos periféricos, que também estão ligados ao barramento único. O problema principal deste projeto é que o barramento torna-se um gargalo, considerando que todas as unidades de processamento devem utilizá-lo para acessar a memória global e os dispositivos periféricos. Portanto, tal barramento restringe a escalabilidade da arquitetura UMA. Esta arquitetura é considerada como multiprocessada com memória compartilhada hierárquica devido a topologia atual do núcleo projetado dentro destas máquinas.

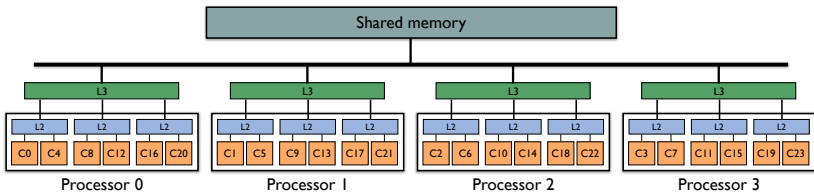


Figura 6: A arquitetura UMA (RIBEIRO et al., 2009)

Plataformas UMA atuais apresentam uma topologia complexa, com múltiplos processadores *multicore* e memórias *cache*. A Figura 6 apresenta uma plataforma *multicore* do tipo UMA. Podemos observar que a máquina possui quatro processadores, cada um contendo seis núcleos. Considerando-se o subsistema de memória, a máquina possui dois níveis de memória *cache* compartilhada. Cada par de núcleos compartilha uma memória *cache* L2 e cada processador uma memória *cache* L3.

A memória principal é compartilhada entre todos os núcleos da

máquina através de um único barramento. Mesmo que estas máquinas possuam um acesso uniforme da memória partilhada, é importante levar em consideração a topologia ao mapear os processos de aplicação/*thread*. Devido à organização hierárquica de núcleos, processadores e memórias *cache*, o tempo de comunicação entre as unidades de processamento podem variar, dependendo da distância entre eles (MEI et al., 2010)(CRUZ; ALVES; NAVAU, 2010). Por exemplo, na máquina apresentada na Figura 6, a hierarquia de memórias *cache* pode ser explorada para reduzir o tempo de comunicação entre um grupo de *threads*. Tal grupo pode ser colocado dentro do mesmo processador, evitando a comunicação entre processadores.

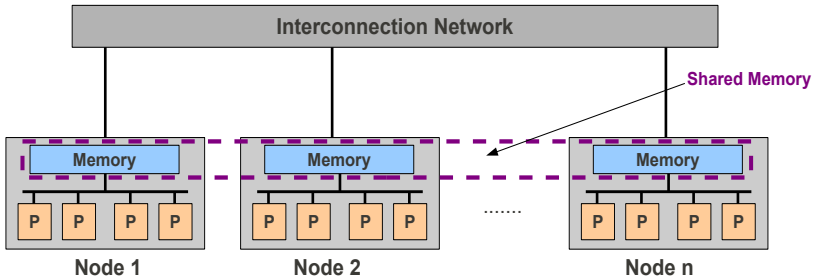


Figura 7: A arquitetura NUMA (RIBEIRO et al., 2009)

A plataforma NUMA (*Non-Uniform Memory Access*) é um sistema multiprocessado em que os elementos de processamento são servidos por vários bancos de memória, distribuídos fisicamente através da plataforma. Embora a memória seja distribuída fisicamente, esta é vista pelas unidades de processamento da máquina como uma única memória compartilhada. Nestas máquinas, o tempo gasto para acessar os dados é condicionada pela distância entre o processador e o banco de memória na qual os dados estão fisicamente alocados (CARISSIMI, 1999).

Arquiteturas NUMA são geralmente projetadas com memórias *cache*, a fim de reduzir penalidades de acesso à memória. Devido a isso, algum suporte para garantir a coerência de *cache* para as unidades de processamento é implementado nas plataformas NUMA atuais, levando a plataformas NUMA de cache coerente denominadas ccNUMA. Uma das vantagens da arquitetura NUMA é que ela combina uma boa escalabilidade de memória com uma característica de programação fácil. Nessas máquinas, uma rede de interconexão eficiente e especializada

fornece suporte ao elevado número de unidades de processamento e memórias de grande porte. Graças ao único espaço de endereçamento global, os programadores podem utilizar modelos de programação de memória compartilhada para desenvolver aplicativos paralelos para estas máquinas.

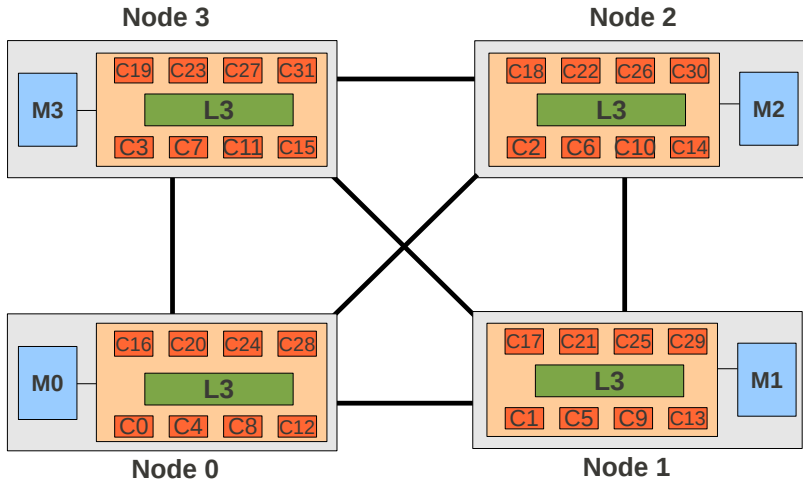


Figura 8: A arquitetura NUMA atual (RIBEIRO et al., 2009)

As Figuras 7 e 8 mostram esquemas representando duas máquinas NUMA. O esquema representado na Figura 7 apresenta as máquinas NUMA clássicas dos anos 80 enquanto a Figura 8 descreve uma máquina NUMA atual com *chips multicore*. Podemos observar em ambas as figuras que as máquinas NUMA são organizadas em múltiplos nós conectados por uma rede de interconexão. Cada nó é geralmente composto por várias unidades de processamento (*monocore* ou *multicore*) e bancos de memória (RIBEIRO et al., 2009).

2.2.6 Manycores

Até recentemente, as capacidades de plataforma de *High Performance Computing* (HPC) foram avaliadas quase que exclusivamente com base na sua velocidade de processamento bruto. No entanto, um dos aspectos que dificultam a busca de desempenho cada vez maior é o excessivo consumo de energia. Por essa razão, a eficiência energética de

plataformas HPC tem se tornado, em alguns contextos, tão importante quanto o seu desempenho bruto. A busca de alternativas para reduzir o consumo atual de energia não se limita à comunidade científica de HPC (RAJOVIC et al., 2013). Recentemente, surgiram os processadores *manycore*, como uma nova classe de *chips* altamente paralelos. Tiler Tile-GX (MORARI et al., 2012), Kalray MPPA-256 (DINECHIN et al., 2013), Adapteva Epiphany-IV (VARGHESE et al., 2014), Intel Single-Chip Cloud Computer (SCC) (TOTONI et al., 2012) e Xeon Phi são exemplos de tais processadores, fornecendo até centenas de núcleos de processamento autônomos que podem ser utilizados para alcançar paralelismo tanto de dados quanto de tarefas.

Esta distinta característica os diferencia dos processadores gráficos (GPUs). Enquanto alguns processadores *manycore* podem apresentar uma maior eficiência energética do que processadores *multicore* de uso geral, suas diferenças arquiteturais implicam em um desafio quanto ao desenvolvimento de aplicações científicas paralelas (VARGHESE et al., 2014)(CASTRO et al., 2014).

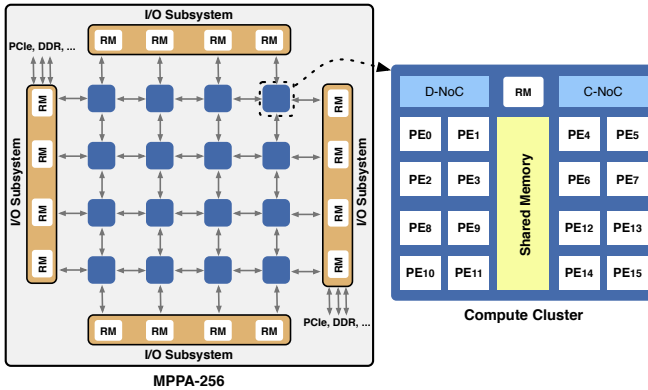


Figura 9: Visão simplificada de um MPPA-256 (CASTRO et al., 2014)

A Figura 9 mostra a visão geral da arquitetura do MPPA-256. Ele possui dois tipos de núcleos: *Processing Elements* (PE) e de *Resource Managers* (RM). Embora RMs e PEs implementem a mesma arquitetura de *Very long instruction word* (VLIW), eles possuem propósitos diferentes: PEs são dedicados a executar *threads* do usuário (uma *thread* por PE) em modo não-interruptível e não-preemptivo. Por outro lado, RMs executam rotinas do *kernel* e serviços de interfaces de Network-on-Chip (NoC). PEs e RMs são agrupados dentro de *clusters*

de computação e subsistemas de I/O. Cada *cluster* de computação possui 16 PEs, 1 RM e uma memória local compartilhada de 2 MB, o que cria uma interligação com alta largura de banda e taxa de transferência entre PEs. O sistema não possui nenhum protocolo de coerência de cache entre PEs, mesmo entre aqueles no mesmo cluster de computação.

2.3 ESCALONAMENTO DE TAREFAS

Segundo Tanenbaum e Woodhull (2006), o escalonador é responsável por decidir qual tarefa irá executar primeiro, caso haja várias tarefas prontas para executar competindo pelo uso do processador. O escalonador é também responsável por: decidir qual é o processador mais adequado para executar cada tarefa, decidir qual é o intervalo de tempo que cada tarefa executará em cada processador e alocar os recursos necessários para cada tarefa e disparar a sua execução. De forma ideal, um escalonador deveria garantir que as tarefas executassem utilizando ao máximo os recursos disponíveis e terminassem no menor tempo possível, sempre respeitando as restrições de tempo ou outras políticas aplicadas às tarefas.

Na literatura, encontram-se diversos algoritmos de escalonamento que se adaptam a diferentes tipos de problemas e sistemas. Dentre eles, há alguns que se destacam pela facilidade de implementação, adaptabilidade e desempenho, tais como (TANENBAUM; WOODHULL, 2006):

- ***First In First Out (FIFO)***: algoritmo de escalonamento não preemptivo, no qual as tarefas são executadas pelo processador na mesma ordem em que o solicitam. Como vantagens deste algoritmo, pode-se citar a facilidade de entendimento e implementação, além de sua imparcialidade. Como desvantagens pode-se mencionar a sensibilidade à ordem de chegada das tarefas e o aumento do tempo médio de espera no caso de tarefas grandes chegarem primeiro na fila.
- ***Last In First Out (LIFO)***: este algoritmo funciona de maneira simples, a última tarefa a entrar na fila será a primeira a ser executada.
- ***Shortest Job First (SJF)***: o SJF também é um algoritmo não preemptivo, o qual assume que os tempos de execução das tarefas são conhecidos antecipadamente. Caso haja várias tarefas na fila de entrada, de igual importância a serem executadas, o escalonador irá selecionar a tarefa mais curta primeiro.

- **Shortest Remaining Time Next (SRT)**: no algoritmo SRT o escalonador escolhe a tarefa cujo tempo de execução restante é o mais curto. Ou seja, ao chegar uma nova tarefa o seu tempo será comparado com o tempo que resta da tarefa atual. Caso a nova tarefa necessite de menos tempo para terminar do que a tarefa atual, esta será suspensa e a nova tarefa será executada.
- **Round-Robin (RR)**: o algoritmo RR opera em sistemas iterativos, realizando um rodízio entre as tarefas da seguinte maneira: cada tarefa possui um intervalo de tempo denominado *quantum*, durante o qual ela pode ser executada. Caso ela esteja em execução e seu *quantum* termine, esta será reinserida no final da fila liberando o processador.
- **Earliest Deadline First (EDF)**: esse algoritmo dá prioridade de execução à tarefa que possui o *deadline* mais próximo de expirar, ordenando as tarefas com base em seu *deadline*.
- **Backfill First Fit (BFF)**: o algoritmo BFF funciona de maneira parecida com o FIFO, mas quando não há recursos suficientes para a execução da primeira tarefa da fila, o restante da fila é examinado para encontrar a primeira tarefa que possa ser executada com os recursos e tempo disponíveis.
- **Backfill Best Fit (BBF)**: o algoritmo BBF funciona de maneira parecida com o FIFO e o BFF, mas quando não há recursos suficientes para a execução da primeira tarefa da fila, o restante da fila é examinado para encontrar a tarefa que melhor se encaixa para ser executada com os recursos e tempo disponíveis

Considerando desafios e estratégias conhecidas para o processo de escalonamento de recursos em grades computacionais, existem alguns algoritmos específicos mais utilizados em tais ambientes:

- **Workqueue (WQ)**: WQ é um agendador livre de conhecimento, ou seja, não necessita de qualquer tipo de informações sobre agendamento de tarefas. Sempre que um recurso fica disponível, uma tarefa é escolhida aleatoriamente para ser apresentada para a execução. O objetivo é que um maior número de tarefas seja atribuído a máquinas mais rápidas, deixando máquinas mais lentas para executar cargas mais leves. No entanto, se uma tarefa que requer uma grande capacidade computacional é atribuída a um processador mais lento perto do fim da execução da aplicação,

a conclusão da aplicação será adiada até que a tarefa seja concluída (SILVA; CIRNE; BRASILEIRO, 2003).

- **Workqueue with Replication (WQR)**: WQR é semelhante ao WQ; tarefas são enviadas para executar nas máquinas que estão disponíveis. No momento em que uma máquina termina a execução de uma tarefa, recebe uma nova tarefa. A diferença entre WQ e WQR ocorre quando uma máquina se torna disponível e não há outras tarefas na fila para serem executadas. Neste momento, o WQR inicia a replicação de tarefas que ainda estão em execução. Uma vez que a tarefa original ou uma de suas réplicas terminar a execução, as outras são interrompidas (SILVA; CIRNE; BRASILEIRO, 2003).
- **Sufferage (Suff)** (CASAVANT; KUHL, 1988): Este algoritmo determina o quanto cada tarefa seria prejudicada se não escalonada para o processador que a execute de forma mais eficiente. O valor *Sufferage* de cada tarefa é a diferença entre o melhor e o segundo melhor tempo de conclusão (CT) entre todos os processadores na grade. A tarefa com o maior valor de *Sufferage* terá prioridade de execução. O valor de CT é dado pela fórmula:

$$CT = \text{Time to Become Available (TBA)} + \text{Task Cost} \quad (2.1)$$

Onde TBA é o momento em que o *host* se torna disponível e *Task Cost* é dado pela fórmula:

$$\text{Task Cost} = \frac{\left(\frac{\text{Task Size}}{\text{Host Speed}}\right)}{(1 - \text{Host Load})} \quad (2.2)$$

onde:

- *Task Size*: é o tempo necessário para um *host* com *Host Speed* = 1 completar a tarefa quando *Host Load* = 0.
 - *Host Speed*: representa a velocidade da máquina.
 - *Host Load*: representa a fração de CPU do *host* que não está disponível para a aplicação, ou seja, a fração de processador que está sendo usada por outras tarefas. O *Host Load* varia com o tempo, dependendo da carga de trabalho do *host*.
- **XSufferage (XSuff)**: *XSufferage* é uma modificação do *Sufferage*. Sua principal diferença é o método utilizado para calcular o

Sufferage. *XSufferage* considera a transferência dos dados de entrada da tarefa durante o cálculo dos tempos de execução. Dessa forma, ele utiliza as informações relacionadas à CPU e ao tempo estimado de execução da tarefa usado pelo *Sufferage* mais a largura de banda disponível na rede que conecta os recursos. Para que o recurso mais rápido e com melhor conexão de rede não receba todas as tarefas, o *XSufferage* considera somente os recursos livres no momento em que vai escalonar uma tarefa.

- ***Dynamic Fastest Processor to Largest Task First (DFPLTF)***: Esta é a versão dinâmica do algoritmo estático FPLTF (MENASCÉ et al., 1995). O DFPLTF precisa de três elementos de informação adicional para agendamento de tarefas: *Task Size*, *Host Load* e *Host Speed*. Quando o algoritmo inicia, o TBA do *host* é iniciado em zero e as tarefas são ordenados por tamanho em ordem decrescente. Assim sendo, a maior tarefa é a primeira a ser alocada para o *host* que provê o menor CT. Uma vez que a tarefa é atribuída a um *host*, o valor de TBA relativo a este *host* é incrementado com o *Task Cost*. As tarefas são atribuídas até que todos os *hosts* na grade estejam em uso. Depois disto, a execução da tarefa é iniciada. Quando a tarefa for concluída, todas as outras tarefas que não estão executando são alocadas novamente até que todas as máquinas continuem em uso. Este processo é repetido até que todas as tarefas estejam concluídas.

Segundo Reis (2005), o desenvolvimento de algoritmos de escalonamento deve ser focado em um conjunto de aplicações específicas, pois se não houver um conhecimento dos detalhes das aplicações a serem escalonadas, o algoritmo pode influenciar negativamente nos resultados. Como um ambiente de grade computacional possui algumas características especiais, tais como: grande quantidade e heterogeneidade de recursos e desempenho dinâmico, o escalonamento nesse tipo de ambiente torna-se um desafio. Por isso, como dito anteriormente, conhecer as aplicações com as quais se irá trabalhar, os recursos disponíveis e as características dos mesmos, pode ser um fator crucial para o desenvolvimento de algoritmos de escalonamento que consigam utilizar de maneira eficiente o alto poder computacional das grades.

2.4 SIMULAÇÃO

A simulação é uma ferramenta adequada para analisar algoritmos de sistemas distribuídos em larga escala de recursos heterogêneos. Ao contrário de se utilizar sistemas reais, a simulação evita a sobrecarga de coordenação de recursos reais. A simulação é igualmente eficaz no trabalho com grandes problemas hipotéticos, que exigem envolvimento de grande número de usuários ativos e de recursos (BUYA; MURSHED, 2002)

A motivação para a utilização de simulação ao invés de utilizar diretamente uma plataforma de teste real de grade (especialmente nos modelos de análise e algoritmos em fases iniciais), pode ser definida a partir dos seguintes fatores (MURSHED; BUYA et al., 2002) (BUYA; SULISTIO, 2008):

- A criação de uma plataforma de teste real de grade é demorada, de custo elevado, depende de recursos intensivos e é limitada na maior parte a alguns ambientes de área local;
- Uma plataforma de teste real não fornece um ambiente controlável e de repetição para experimentação e avaliação de estratégias de escalonamento;
- A simulação permite analisar tanto os atuais, como os novos modelos energéticos e algoritmos de escalonamento;
- A análise de novos modelos e algoritmos requer um grande número de testes dependentes de recursos disponíveis. A simulação permite uma análise abrangente dos modelos e algoritmos sem a necessidade de gastos com recursos físicos específicos.

A simulação tem sido pesquisada e aplicada com sucesso para modelar processos, aplicações e objetos do mundo real. Ela permite o estudo de vários assuntos, tais como a viabilidade, o comportamento e o desempenho, sem construir o sistema real, poupando assim um tempo precioso, custo e esforço. Uma simulação pode ser ajustada de acordo com vários cenários possíveis. Os resultados obtidos a partir da simulação indicam como o sistema real se comporta, permitindo, assim, que os pesquisadores compreendam e melhorem seus projetos antes de colocarem os esforços na implementação real propriamente dita (SULISTIO; YEO; BUYA, 2004)(BUYA; SULISTIO, 2008).

Em Franco (2011) é apresentado um comparativo entre características de simuladores voltados à grade. A Figura 10 apresenta as principais características das ferramentas.

	Brincks	GridSim	MicroGrid	OptorSim	SimGrid
Tipo de ferramenta	Simulador	Simulador	Emulador	Simulador	Simulador
Linguagem de implementação	Java	Java	C	Java	C / Java
Plataformas suportadas	Qualquer S.O. com suporte a JVM	Qualquer S.O. com suporte a JVM	Linux/Alpha em um <i>Grid</i> Globus	Qualquer S.O. com suporte a JVM	Linux, MacOS, Windows
Interface	Modo texto	Modo texto e GUI (limitada)	Modo texto	Modo texto e GUI	Modo texto
Framework de programação	Orientada a objeto	Orientada a objeto	Estruturada	Orientada a objeto	Estruturada
Ambiente de projeto	Linguagem	Biblioteca	Linguagem	Biblioteca	Biblioteca
Simulação	Estática, discreta, determinística	Estática, discreta, determinística	Dinâmica, contínua, determinística	Dinâmica	Estática, discreta, determinística
Mecanismo de simulação	Sequencial	Sequencial	Paralelo	Multitarefa	Sequencial
Arquitetura do <i>grid</i>	Definida pela ferramenta	Definida parcialmente pelo usuário	Definida pelo usuário de acordo com o <i>Globus Grid</i>	Definida pela ferramenta	Definida pelo usuário
Licença	Opensource	Opensource	Opensource	Opensource	Opensource

Figura 10: Comparativo de Simuladores de Grade (FRANCO, 2011)

No presente trabalho, utiliza-se a ferramenta de simulação de aplicações em ambientes distribuídos heterogêneos SimGrid (LEGRAND; MARCHAL; CASANOVA, 2003), criada em 1999 por Henri Casanova. A construção da ferramenta deu-se pela necessidade de se utilizar simulação, ao invés de experimentos reais, no estudo de algoritmos de escalonamento para aplicações científicas paralelas. Como o objetivo inicial do SimGrid era trabalhar com escalonamento de tarefas, existe uma facilidade para estudar estratégias de escalonamento utilizando a ferramenta. Como motivações para utilizar essa ferramenta, citam-se:

- Especificação dos recursos de computação disponíveis, juntamente com o poder computacional de cada máquina em *Floating-point Operations Per Second* (FLOPS);
- *Links* que conectam os nós do sistema, juntamente com a largura de banda e suas latências;
- Roteamento entre os nós, no qual são especificados o nó de origem, o nó de destino e os *links* de conexão que os unem;
- Simulação determinística, ou seja, repetindo a mesma simulação os resultados retornados serão sempre os mesmos. Não existindo assim, por parte do simulador, uma aleatoriedade nos experimentos.
- Está disponível para ambientes Linux, Windows e MacOS;

- Uma comunidade responsiva a questões referentes ao desenvolvimento de códigos.

Segundo Franco (2011), apesar do SimGrid ser uma ferramenta que auxilia no estudo de algoritmos de escalonamento em uma grade computacional, ele não oferece políticas internas de escalonamento de tarefas. Dessa forma, a implementação dos algoritmos de escalonamento deve ser feita pelos próprios usuários. Com o objetivo de criar um ambiente amigável e que facilite o trabalho dos pesquisadores da área de escalonamento de tarefas foi desenvolvida a biblioteca LIBTS(FRANCO, 2011). A LIBTS é desenvolvida em linguagem C e implementa os seguintes algoritmos de escalonamento: FIFO, LIFO, RR, SJF, WQ, WQR, Sufferage, XSufferage e DFPLTF.

2.5 CONSIDERAÇÕES SOBRE O CAPÍTULO

Nesse capítulo foram abordados conceitos sobre as infraestruturas de sistemas distribuídos existentes. *Cluster* é um conjunto de máquinas que podem ou não ser dedicadas, geralmente homogêneas, de um único domínio e, normalmente, interligadas por redes de alto desempenho. Já o *Multi-Cluster* é um conjunto de vários *clusters* interligados por rede WAN. Com isso, possui vários domínios. A grade computacional é composta por máquinas heterogêneas geograficamente distribuídas e, normalmente, não dedicadas. A nuvem computacional pode ser considerada uma junção de tecnologias existentes, porém utilizadas de maneira diferenciada, permitindo o acesso sob demanda a recursos e serviços através da Internet. Este trabalho foca em uma proposta aplicada a uma infraestrutura de um ambiente de nuvem computacional criado através do simulador SimGrid. O SimGrid possui algumas características que o tornam o ideal para testar nossa abordagem tais como: possibilita a criação personalizada e detalhada de um ambiente de grade; oferece simulações do tipo: estática, discreta e determinística; permite a implementação do código em linguagem C e Java; é amplamente utilizado pela comunidade científica tornando-se possível o reaproveitamento de códigos fontes.

3 TRABALHOS RELACIONADOS

Neste capítulo é apresentado o estado da arte em técnicas para a redução do consumo de energia em sistemas computacionais heterogêneos de alto desempenho, como grades e nuvens. Ao final deste, é apresentada uma tabela com um comparativo entre os trabalhos apresentados e o trabalho desenvolvido. O objetivo de apresentar o estado da arte é mostrar que a maioria dos trabalhos pesquisados a respeito de estratégias de economia de energia em grades e nuvens computacionais apresentam soluções que utilizam controle de recursos ociosos e escalonamento de tarefas para obter eficiência energética.

3.1 INTRODUÇÃO

De acordo com Mämmelä et al. (2012), a fim de garantir a eficiência energética em um ambiente de grade computacional é preciso dar atenção aos seguintes fatores:

- Ajuste dinâmico da frequência e da voltagem do processador, *Dynamic Voltage and Frequency Scaling* (DVFS);
- Desligamento de componentes de *hardware* de baixa utilização;
- Nivelamento de potência;
- Gestão térmica.

A técnica de DVFS é utilizada para controlar a potência do processador, pois segundo Mämmelä et al. (2012), o consumo de energia do processador representa uma parcela significativa da energia total consumida pelo sistema. Os autores também citam que alguns servidores ou seus componentes poderiam ser desligados, ou então, poderiam operar em um estado de baixo consumo de energia, sempre atendendo às necessidades das aplicações de entrada. Porém, devido ao fato desta solução ser dependente da carga de trabalho, o desafio, de acordo com Mämmelä et al. (2012), seria identificar o momento certo de desligar componentes e como fornecer um valor adequado de desaceleração de trabalho.

Técnicas de gestão térmica são utilizadas para gerenciar a elevação das temperaturas, o que pode impactar negativamente na confiabilidade do sistema e pode causar o aumento dos custos com resfriamento. No

trabalho desenvolvido por Liu e Zhu (2010), a carga de trabalho do sistema é ajustada de acordo com um limiar de temperatura pré-definida. Caso a temperatura de um servidor fique acima do limite, sua carga de trabalho será reduzida. Nivelamento de potência se refere a definição de limites de potência em servidores. Esta estratégia previne aumentos repentinos no fornecimento de potência e mantém o consumo total sobre um valor especificado (FAN; WEBER; BARROSO, 2007).

3.2 CONTROLE DE RECURSOS OCIOSOS

Essa estratégia consiste em alterar o estado de recursos computacionais disponíveis ociosos para que não permaneçam consumindo energia de maneira desnecessária. A seguir, são apresentados alguns trabalhos que utilizam tal estratégia

3.2.1 Ponciano e Brasileiro

Ponciano e Brasileiro (2010) define grade oportunista como uma infraestrutura computacional distribuída que utiliza ciclos computacionais ociosos de recursos computacionais geograficamente distribuídos, e que este tipo de sistema é ideal para execução de aplicações do tipo *Bag of Tasks* (BoT), aplicações paralelas que podem ser divididas em um grande número de tarefas independentes sem necessidade de comunicação entre si.

No trabalho de Ponciano e Brasileiro (2010), foram investigados três aspectos de uma grade oportunista de acordo com duas métricas: economia de energia e *makespan* de tarefas (diferença entre os tempos de início e fim de execução de uma tarefa). Foram avaliados os seguintes aspectos:

- **Estratégias de hibernação:** utilizadas para reduzir consumo de energia da grade durante períodos de inatividade;
- **Estratégias de despertar:** utilizadas para selecionar um conjunto de recursos a fim de satisfazer uma demanda de tarefa;
- **Estratégias de escalonamento:** determinam em qual máquina disponível cada tarefa deverá ser executada.

O ambiente utilizado para os experimentos foi simulado. Segundo Ponciano e Brasileiro (2010), medir a energia real consumida

por uma grade oportunista é um desafio, uma vez que os recursos computacionais são heterogêneos, dispersos geograficamente, e requisitados de maneira oportunista. Devido à inexistência de um simulador de grade oportunista que suportasse a simulação de estratégias de escalonamento para tarefas do tipo BoT, estratégias de hibernação e que medisse o consumo de energia, os autores optaram pelo desenvolvimento de um simulador baseado em eventos. O simulador desenvolvido oferece a possibilidade de reproduzir estratégias de escalonamento de tarefas do tipo BoT e estratégias de hibernação.

Em seus experimentos foram avaliadas duas estratégias de hibernação: *standby* e *hibernate*. Seus resultados mostraram que ambos podem proporcionar uma economia substancial no consumo de energia nas grades oportunistas. Em cenários simulados, a economia pode chegar a mais de 80%, quando comparado com a energia consumida quando os recursos não utilizam a estratégia de hibernação. Mais importante ainda, essas economias são atingidas sem uma grande penalidade associada com o aumento do *makespan* das tarefas.

Também foram avaliadas duas estratégias de despertar: *Energy-Aware* (EA) e *Most Recently Sleeping* (MRS), as quais apresentaram resultados semelhantes em termos de *makespan* de tarefas. No entanto, em cenários de baixa contenção de recursos, a estratégia EA apresentou melhores resultados de economia de energia do que a estratégia MRS. Por fim, foram avaliadas quatro políticas de escalonamento de tarefas: FCFS, FPLT, MEEF e MEELT. Estas estratégias consideram as diferentes características de recursos, bem como a carga de trabalho da grade. Porém, não foram observadas diferenças significativas em relação à economia de energia e *makespan* entre as quatro políticas de escalonamento.

3.2.2 Mämmelä et al.

Em (MÄMMELÄ et al., 2012) é apresentado um escalonador consciente de energia para *datacenters* HPC. O escalonador se comunica com os sistema de gerenciamento de recursos do *datacenter*. O escalonador é um local natural para a tomada de decisões conscientes de consumo energético, tendo em vista que ele é responsável por decidir em que máquinas cada tarefa será executada. O escalonador proposto suporta três algoritmos de escalonamento comumente utilizados: FIFO (ou FCFS), BFF e BBF (*Backfilling*).

Seus experimentos foram realizados em duas etapas: em um

ambiente simulado (primeira etapa) e em um ambiente real (segunda etapa). Na primeira etapa os autores concluíram que o escalonador proposto pode ser aplicado a ambientes de *datacenters* HPC sem modificações no *hardware*. O resultado das simulações mostrou reduções no consumo de energia variando entre 6% e 16%, dependendo da utilização do sistema e algoritmo de escalonamento utilizado. As reduções no consumo de energia foram obtidas sem um aumento drástico do tempo de espera médio das tarefas.

Com relação a segunda etapa (ambiente real), foi alcançada uma economia de 6,3% com um aumento no tempo de execução das tarefas de 0.63%. Os algoritmos E-BFF e o E-BBF, mostraram-se mais eficientes do que o E-FIFO, pois diminuem o tempo de espera das tarefas por explorar o preenchimento dos nós ociosos com a execução de tarefas menores, o que não ocorre no E-FIFO.

3.2.3 Montes et al.

O trabalho de (FERNÁNDEZ-MONTES et al., 2012) realiza um estudo de políticas operacionais em uma grade computacional a fim de otimizar o uso de recursos. O estudo é feito a partir de informações de uso extraídas da grade Grid'5000 (CAPPELLO et al., 2005). Neste trabalho são considerados, para os inúmeros nós que compõem a grade, os seguintes estados:

- **On:** quando o recurso está executando uma tarefa. A potência considerada foi de 108 watts;
- **Off:** significa que o recurso está desligado e portanto, não está ocupado com tarefas. A potência considerada foi de 5 watts;
- **Idle:** significa que o recurso está ligado esperando pelo recebimento de tarefas para executar. A potência considerada foi de 50 watts;
- **Booting:** significa que o recurso está passando do estado de Off para On. A potência considerada foi de 110 watts;
- **Shutting:** quando um recurso é desligado a partir de On ou Idle. A potência considerada foi de 110 watts.

A proposta do trabalho estabelece um conjunto de políticas que giram em torno do desligamento ou não de máquinas em estado ocioso. A decisão é tomada mediante análise do custo de alteração do estado.

Os testes foram realizados em um ambiente que simulou o Grid'5000, denominado *Grid'5000 toolbox*. O simulador possibilita o usuário definir inúmeros parâmetros incluindo: horário do início da simulação, horário do fim da simulação, localização, política de energia e política de arrançamento de máquinas.

3.2.4 Laszewski et al.

O trabalho de Laszewski et al. (2009) foca na implementação de um algoritmo de escalonamento consciente de consumo para computação em *clusters* de alto desempenho, onde as máquinas virtuais são fornecidas de maneira dinâmica para a execução de tarefas do *cluster*. É proposto um novo algoritmo de escalonamento para *cluster* com o objetivo de reduzir a dissipação de energia do processador, através de variação de frequências do processador sem aumentar drasticamente o tempo total de execução da máquina virtual. Este algoritmo é implementado em um simulador de *clusters* com DFVS habilitados.

O modelo e algoritmo propostos foram implementados e aplicados ao projeto OpenNebula em um *cluster multicore*. Os componentes principais do OpenNebula aceitam solicitações de usuários por meio de uma interface, e então alocam as máquinas virtuais dentro do *cluster*. O desenvolvimento modular do OpenNebula permite a substituição do algoritmo de alocação de VMs, o que o torna ideal para os testes.

3.3 ESCALONAMENTO VOLTADO A EFICIÊNCIA ENERGÉTICA

3.3.1 Montero et al.

Montero, Huedo e Llorente (2003) analisaram a relevância da proximidade de recursos no processo de seleção de recursos para reduzir o custo na etapa de *file staging* (etapa de preparação de dados para transferência). Também estudaram migração oportunista quando um novo recurso se torna disponível na grade. Nessa situação, o desempenho do novo *host*, os tempos de execução das aplicações restantes, e a proximidade do novo recurso aos dados necessários são considerados fatores críticos para decidir se a migração da tarefa é viável e rentável. O novo processo de seleção de recursos apresentado por Montero, Huedo e Llorente (2003) considera tanto desempenho quanto proximidade

de dados de recursos computacionais. Em particular, são considerados os seguintes cenários no estágio de seleção:

- O tempo computacional estimado no *host* candidato a ser avaliado quando a tarefa é enviada a partir do cliente ou migrada a partir do *host* de execução atual;
- A proximidade entre o *host* candidato que está sendo avaliado e o cliente é considerada com intuito de reduzir os custos de envio da tarefa, monitoramento e *file staging*(processo de preparação de arquivos para transferência);
- A proximidade entre o *host* candidato que está sendo avaliado e um servidor de arquivos remoto é também considerada para reduzir os custos de transferência, quando alguns arquivos de entrada ou saída, são armazenados em tal servidor;
- A proximidade entre o *host* candidato que está sendo avaliado e o *host* de execução atual também é considerada para reduzir a sobrecarga de migração.

As estratégias apresentadas são implementadas sobre o *Framework* GridWay (HUEDO; MONTERO; LLORENTE, 2002). GridWay é um *framework* experimental baseado em Globus, que permite uma execução mais fácil e eficiente de tarefas em um ambiente de grade dinâmica que não armazena histórico de execuções. O núcleo do *framework* GridWay é um agente de submissão pessoal que executa automaticamente as etapas envolvidas no envio de tarefas: seleção do sistema, preparação do sistema, submissão, acompanhamento de execução, migração de tarefas e de finalização de tarefas.

O usuário interage com o *framework* através de um gerenciador de solicitações, que lida com as solicitações do cliente (apresentar, interromper, parar, recomeçar, entre outros) e os encaminha para o gerente de expedição. O gerenciador de recursos é acionado periodicamente a cada intervalo de escalonamento, o qual busca por *matches* de execuções pendentes aos recursos da grade (obtidos através do módulo seletor de recursos). Uma vez que um trabalho é atribuído a um recurso, um gerenciador de submissão e um monitor de desempenho são instanciados para acompanhar a execução.

3.3.2 Nesmachnow et al.

Nesmachnow et al. (2013) introduziram uma nova formulação para o problema de alocação de recursos para sistemas de grades computacionais *multicores* heterogêneas com foco na redução do consumo de energia e do *makespan*. Eles utilizaram um modelo de dois níveis em que um agente de gerenciamento recebe todas as tarefas do usuário e as aloca nos recursos disponíveis pertencentes a diferentes provedores locais.

Este trabalho tem como objetivo avaliar vinte heurísticas multiobjetivo altamente eficientes. Estas heurísticas são modificações e combinações em diferentes formas de heurísticas simples presentes na literatura voltadas a otimização de *makespan* de tarefas independentes.

Em contraste com as técnicas de multiobjetivos existentes, estas novas heurísticas são adequadas para serem implementadas em um sistema real, por fornecerem uma solução única e precisa com uma resposta rápida, superando as desvantagens das abordagens agregadoras e de Pareto. As heurísticas propostas fornecem soluções com o mínimo consumo de energia para diferentes níveis de *makespan*. Em seguida, o projetista do sistema pode utilizar a heurística mais adequada entre as apresentadas para o seu caso específico.

Neste trabalho, as heurísticas propostas foram aplicadas a um ambiente real. Os resultados mostraram que a heurística MaxMIN obteve o melhor desempenho com relação ao *makespan*. Com relação ao consumo de energia, as melhores heurísticas foram SuffMIN, MaxMIN e MinMIN.

3.3.3 Teodoro et al.

Teodoro, Carmo e Fernandes (2013) apresentaram uma solução de gerenciamento eficiente de energia para grades computacionais utilizando escalonamento consciente de energia. Seu principal objetivo é reduzir o consumo de energia para tarefas em execução sem afetar de maneira significativa o desempenho da aplicação.

A solução apresentada é baseada no algoritmo de alocação chamado LECSA, que tenta atribuir tarefas com maior consumo de energia a *hosts* com melhores valores de eficiência energética, desligando *hosts* não utilizados quando necessário. Os experimentos foram realizados em um ambiente simulado e o algoritmo LECSA foi comparado a outros algoritmos clássicos de escalonamento. Os resultados mostraram que o

algoritmo LECSA atingiu uma maior redução no consumo de energia, uma vez que os algoritmos clássicos não consideram consumo de energia em seu processo de escalonamento.

3.3.4 Kim et al.

Muitos estudos em *clusters* foram desenvolvidos para dar suporte a SLAs entre usuários e provedores de recursos. SLAs definem as regras acordadas entre provedores de serviços e consumidores e incluem parâmetros de *Quality of Service* (QoS), como por exemplo prazos de entrega. Embora seja importante reduzir o consumo de energia do sistema, parâmetros de QoS especificados nos SLAs não devem ser violados ou a degradação deve ao menos ser minimizada.

Kim et al. (KIM; BUYYA; KIM, 2007) propuseram dois algoritmos de escalonamento conscientes de energia para aplicações BoT com restrições de prazo em sistemas de *cluster* com DVS habilitado, sendo um deles para políticas de compartilhamento de espaço e outro para políticas de compartilhamento de tempo.

Os algoritmos propostos alteram a voltagem de alimentação dos processadores para minimizar o consumo de energia. Os algoritmos propostos foram desenvolvidos para ambientes *clusters* homogêneos.

3.3.5 Garg e Buyya

Garg e Buyya (2009) abordaram o problema da eficiência energética de grades em nível de meta escalonamento. Em seu trabalhos meta escalonamento é definido como primeiro passo para reserva de recursos, onde recursos e períodos de tempos para execução de tarefas são selecionados, mas sem realizar nenhuma reserva física. Eles examinaram como um meta escalonador de grade pode explorar a heterogeneidade global da infraestrutura da grade para alcançar redução no consumo de energia. Em particular, os autores focaram em projetar um política de meta escalonamento que pode ser facilmente adaptada a meta escalonadores existentes sem modificações na infraestrutura da grade.

Foi proposto o algoritmo de meta escalonamento consciente de heterogeneidade HAMA. O algoritmo trata o problema de escalonamento atribuindo mais carga de trabalho com *deadline* urgente para recursos com maior eficiência energética. Em um segundo passo o algoritmo utiliza DVFS para reduzir ainda mais o consumo de ener-

gia. O algoritmo proposto considera informações sobre recursos globais da grade, tais como eficiência do sistema de resfriamento e eficiência energética da CPU, porém não considera o envio de arquivos de entrada pela rede.

Resultados dos experimentos retratam o efeito de urgência de *deadline* de tarefas sob o consumo de energia, conforme a urgência de conclusão de tarefas aumenta o consumo de energia acompanha seu aumento. O consumo também aumenta em CPUs com menor eficiência energética pois a frequência das CPUs são aumentadas ao máximo para evitar violações de *deadline*.

3.3.6 Castro et al.

Castro et al. (2011) propuseram uma abordagem eficiente em termos de energia para grades oportunistas baseadas em virtualização. A virtualização é uma boa abordagem para fornecer completo isolamento entre os ambientes de computação e do usuário final. São apresentadas duas abordagens quando um *cluster* virtual precisa ser inicializado: (1) escolher máquinas físicas de forma aleatória, ou (2) escolher as máquinas onde os usuários estão trabalhando.

A abordagem de Castro et al. (2011) utiliza UnaGrid, uma infraestrutura oportunista cujo propósito é fornecer mecanismos de virtualização para um ambiente de alto desempenho. Resultados de experimentos apontaram para uma economia de energia. Utilizar recursos de maneira oportunista em infraestruturas legadas possibilitará a indústrias reduzir investimentos em tecnologias e o isolamento de ambos ambientes em um único computador possibilitará uma redução no consumo de energia.

3.4 CONSIDERAÇÕES SOBRE O CAPÍTULO

A Tabela 1 apresenta um comparativo entre os trabalhos descritos anteriormente, ressaltando algumas das suas características principais.

Como pode ser observado na tabela, apenas Montero et al. e Nesmachnow et al. realizaram testes em ambientes reais. Castro et al. é o único dos trabalhos a utilizar virtualização como meio de reduzir consumo de energia em um ambiente de grade. Mämmelä et al., Laszewski et al., Kim et al. e Garg e Buyya fazem uso de técnicas de

	Ano	Ambiente	Tipo de teste	Varição de frequência	Controle de recursos ociosos	Considera rede
Ponciano e Brasileiro	2010	Grade	Simulado	Não	Sim	Não
Mämmelä et al.	2012	Cluster	Simulado	Sim	Sim	Não
Montes et al.	2012	Grade	Simulado	Não	Sim	Não
Laszewski et al.	2009	Cluster	Simulado	Sim	Sim	Não
Teodoro et al.	2013	Grade	Simulado	Não	Não	Sim
Montero et al.	2003	Grade	Ambiente Real	Não	Não	Não
Nesmachnow et al.	2013	Grade	Ambiente Real	Não	Não	Não
Kim et al.	2013	Cluster	Simulado	Sim	Não	Não
Garg e Buyya	2009	Grade	Simulado	Sim	Não	Não
Castro et al.	2013	Grade/Virtualização	Simulado	Não	Não	Não

Tabela 1: Resumo comparativo entre os trabalhos relacionados

variação de frequência de processador para obter redução em consumo energético.

Dos trabalhos apresentados o mais próximo do proposto nesta dissertação seria o de Teodoro et al., que considera o controle de recursos ociosos e propõem um estudo comparativo de algoritmos considerando o consumo de energia como principal ponto de comparação. Porém, as principais diferenças entre o trabalho de Teodoro et al. e o aqui proposto são: (a) a fórmula utilizada para calcular o consumo da transferência de dados pela rede; (b) Teodoro et al. consideram o valor de consumo ocioso da máquina de execução, não considerando portanto a possibilidade de alteração de estado das máquinas; (c) o processo de seleção de recursos desativa máquinas não utilizadas, o que implica em apenas considerar um subconjunto das máquinas disponíveis.

Dos trabalhos identificados na literatura percebe-se que o consumo de energia de dispositivos de rede não é levado em consideração no cálculo da energia consumida pelo ambiente. Outra deficiência identificada nos trabalhos é que os mesmos não consideram variação de estado de recursos em tempo de seleção de recursos, o que limita a quantidade de recursos considerados no processo de seleção.

4 ABORDAGEM ENA (ENERGY AND NETWORK AWARE)

Com a finalidade de explorar soluções de eficiência energética em grades computacionais através de escalonamento de tarefas, desenvolveu-se um modelo para cálculo de consumo de energia e um algoritmo de escalonamento energeticamente eficiente dentro da arquitetura da LIBTS no SimGrid. Este capítulo apresenta tais contribuições e descreve seu funcionamento. Inicialmente, descreve-se a nova estrutura da arquitetura, com a possibilidade de calcular o consumo de energia na execução de aplicações. Em seguida, apresenta-se a métrica utilizada pelo módulo proposto, bem como o próprio módulo responsável pelo cálculo de consumo energético. Por fim, é apresentado um pseudocódigo do algoritmo de escalonamento desenvolvido neste trabalho.

4.1 INTRODUÇÃO

Uma grade é um sistema computacional distribuído de grande escala, que pode ser escalado a ambientes do tamanho da Internet com máquinas distribuídas em várias organizações e domínios administrativos. O surgimento de uma variedade de novas aplicações demanda que as redes suportem mecanismos eficientes de gestão de dados e de recursos. O sistema de gerenciamento de recursos (*Resource Management System* - RMS) é um módulo central para o funcionamento da grade, que é responsável pela gestão do conjunto de recursos disponíveis para a rede, como processadores, largura de banda de rede e armazenamento em disco.

Usualmente, o gerenciador possui um componente de escalonamento, responsável pela seleção de recursos do sistema. Em outros casos, o RMS executa o papel de escalonador. A fim de suportar uma variedade de aplicações de forma eficiente, o RMS deve abordar questões como a tolerância a falhas e estabilidade do sistema. Em uma grade, o conjunto de recursos pode incluir recursos de diferentes provedores, necessitando que todos os provedores de recursos confiem no RMS. Além disso, o RMS é responsável pela manipulação dos diferentes recursos, aderindo a diferentes políticas de uso. A Figura 11 exemplifica a interação do RMS com o conjunto de recursos disponíveis.

Consumidores podem solicitar recursos tanto direta quanto indiretamente à grade, onde consumidores podem ser aplicações, usuários

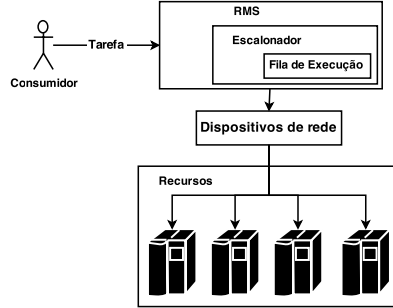


Figura 11: Interação entre RMS e recursos

ou até mesmo outros recursos do ambiente. Tais solicitações de recursos são consideradas tarefas por parte da grade. Na prática, um RMS de grade pode ser obrigado a lidar com tarefas diferentes, utilizando diferentes políticas. Em geral, exigir que o RMS suporte múltiplas políticas pode obrigar mecanismos de escalonamento a tratar um problema de otimização com multicritérios. Idealmente, um escalonador deve garantir que tarefas sejam executadas utilizando recursos disponíveis de maneira ótima e finalizar no menor tempo possível, respeitando limitações de tempo ou outras políticas aplicadas às tarefas (FARIA et al., 2014).

Devido à heterogeneidade e natureza dinâmica da rede, o solicitante do recurso deve estabelecer as exigências a serem cumpridas pelos recursos de destino (processo de descoberta) e os critérios para classificar os recursos selecionados (processo de seleção). Os atributos necessários para a descoberta de recursos e seleção devem ser recolhidos a partir de serviços de informação. Normalmente a descoberta de recursos baseia-se apenas em atributos estáticos, como sistema operacional e arquitetura, enquanto a seleção de recursos baseia-se em atributos dinâmicos, como espaço em disco e carga do processador.

Para ambientes de nuvens o processo é semelhante. O provedor da nuvem (*Cloud Provider*) é responsável pela seleção de recursos do ambiente. Normalmente em um ambiente de nuvem a seleção de recursos é feita considerando as regras estabelecidas entre o provedor e o contratante do serviço (FARIA et al., 2014).

A Figura 12 apresenta os componentes no processo de seleção de recurso. Um consumidor de recursos estabelece contato com o provedor da nuvem e estabelece um acordo de nível de serviço (*Service-level agreement* - SLA) onde os aspectos do serviço, como por exemplo escopo, qualidade, responsabilidades, são formalmente definidos.

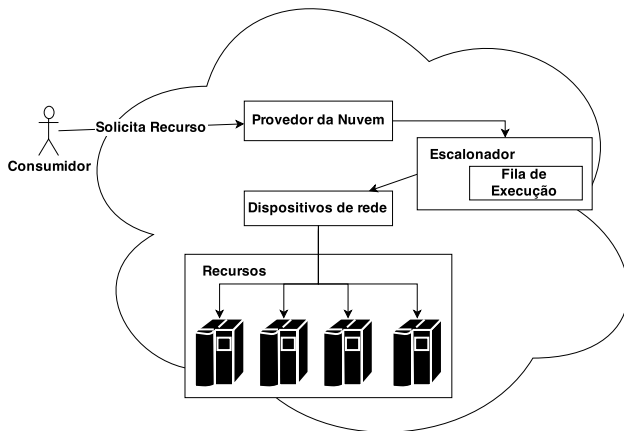


Figura 12: Seleção de recurso em uma nuvem

A partir do SLA o provedor reserva um conjunto de recursos computacionais e os aloca para o usuário. Nestes ambientes se faz uso de técnicas de virtualização, o que possibilita estratégias de economia de recursos como por exemplo migração e consolidação de máquinas virtuais. Esta seleção de máquinas segue diferentes políticas, mas é também realizada por um escalonador, que implementa algum algoritmo de seleção de recursos.

4.2 ARQUITETURA DA PROPOSTA

A fim de estudar uma proposta de eficiência energética em um ambiente genérico de alto desempenho através de escalonamento de recursos, desenvolveu-se um módulo para cálculo de consumo de energia e um algoritmo de escalonamento energeticamente eficiente. A Figura 13 apresenta a alocação da proposta em um ambiente de nuvem computacional.

Após estabelecido o SLA entre consumidor e provedor de serviço, o escalonador utilizaria o novo algoritmo para realizar a seleção do recurso. Para obter informações referentes ao consumo de energia de dispositivos de rede e de recursos computacionais seria necessário que o modelo proposto pudesse acessar tais informações, seja diretamente

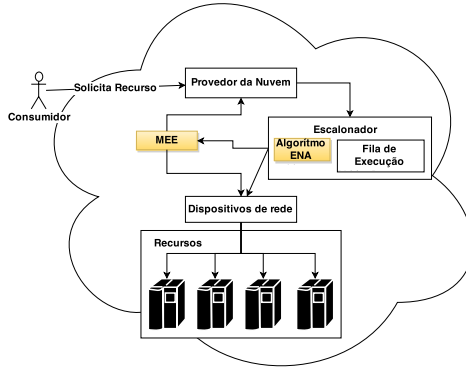


Figura 13: Seleção de recurso em uma nuvem com abordagem ENA

dos recursos ou por intermédio do provedor. O cenário ideal seria obter tais informações sem interferir no funcionamento da nuvem, porém para obter tais informações seriam necessárias modificações no projeto do provedor da nuvem.

Considerando o cenário de seleção de recursos em nível de infraestrutura de uma nuvem, foi desenvolvido um módulo para cálculo de consumo energético. Além disso foi também proposto um algoritmo que utiliza o modelo de cálculo proposto para selecionar recursos. Para validar a abordagem de seleção proposta foi adotada a biblioteca de algoritmos LIBTS (FRANCO, 2011). A Figura 14 apresenta a arquitetura utilizada. As partes em destaque representam o algoritmo de escalonamento voltado a eficiência energética e módulo que implementa o modelo de eficiência energética, desenvolvidos neste trabalho. A figura apresenta a camada dos ambientes de programação. Esta camada fornece diversos ambientes de programação construído sobre um único kernel de simulação. Cada ambiente objetiva um alvo específico e constitui um paradigma diferente. Os componentes dessa camada são (FRANCO, 2011):

- **MSG** (*Simple programming environment*): Foi o primeiro ambiente de programação disponibilizado e é o de uso mais difundido. Usado para modelar aplicações como processos sequenciais concorrentes (Concurrent Sequential Processes), é útil para modelar problemas teóricos e para comparar diferentes heurísticas.

- **SMPI** (*Simulated MPI*): Para simulações de códigos MPI. Simulação do comportamento de uma aplicação MPI usando técnicas de emulação.
- **SimDag**: Ambiente dedicado à simulação de aplicações paralelas, por meio do modelo DAG (Direct Acyclic Graphs). Com este modelo é possível especificar relações de dependência entre tarefas de um programa paralelo.

A camada de simulação do kernel (*Simulation kernel layer*) possui o módulo chamado **SURF** que fornece todas as funcionalidades essenciais para simular uma plataforma virtual. É utilizada como base para a camada de nível superior, e não se destina a usuários finais por ser considerada de muito baixo nível. Uma das principais características do SURF é a capacidade de mudar de forma transparente o modelo utilizado para descrever a plataforma, facilitando muito a comparação dos vários modelos existentes na literatura.

A camada de base (*Base layer*) é constituída pelo **XBT** (*eXtended Bundle of Tools*). O XBT é uma biblioteca portátil que fornece alguns recursos como suporte de registro, suporte de exceção e de configuração. O XBT também abrange as estruturas de dados convenientes: Dynar: *generic Dynamic array*; Fifo: *generic workqueue*; Dict: *generic dictionary*; Heap: *generic heap data structure*; Set: *generic set datatype*; Swag: *O(1) set datatype*.

A implementação das políticas de escalonamento é feita através da programação com a utilização da API em C fornecida pelo simulador. A API permite manipular tipos de dados para recursos e para tarefas. Um recurso é descrito pelo nome, um conjunto de métricas relacionadas a desempenho, traços e constantes. Uma tarefa é descrita pelo nome, custo e estado. Além de funções básicas como criação, inspeção e destruição, são fornecidas funções que descrevem possíveis dependências entre tarefas e funções, para designar tarefas aos recursos (LEGRAND; MARCHAL; CASANOVA, 2003).

4.3 MÉTRICAS

O objetivo geral de métricas de consumo é fornecer uma visão geral da eficiência energética de uma infraestrutura. O modelo de cálculo utilizada neste trabalho é baseado na abordagem proposta por Baliga et al. (BALIGA et al., 2011), que considera a energia por *bit* como uma medida fundamental de consumo. O E_c energia necessária para trans-

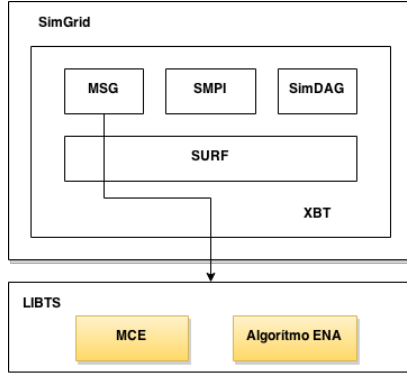


Figura 14: Arquitetura proposta em libts

mitir um *bit* através da rede pode ser expressa como:

$$E_c = 3 \times 3 \times \left(\frac{P_{les}}{C_{les}} + \frac{P_{es}}{C_{es}} + \frac{P_g}{C_g} \right) \quad (4.1)$$

onde P_{les} , P_{es} , e P_g são, respectivamente, a energia consumida por *switches* Ethernet pequenos, *switches* Ethernet regulares, e *routers* e C_{les} , C_{es} e C_g representam as capacidades de cada equipamento em *bits* por segundo. O primeiro fator 3 na Equação 4.1 inclui a necessidade de energia para redundância (com um valor de 2), bem como incorpora a perda de energia de refrigeração (com um valor estimado de 1,5). O segundo fator 3 reflete os cálculos do consumo combinados para os *switches* da rede e rede LAN da grade.

4.4 MODELO DE EFICIÊNCIA ENERGÉTICA (MEE)

O consumo de energia de uma tarefa refere-se à energia gasta pelo ambiente para executar uma tarefa em uma máquina de execução. Além da energia necessária para transportar os *bits* dos arquivos de entrada na rede, as máquinas de execução podem estar em diferentes estados, em diferentes níveis na rede, e a diferentes distâncias do nó com os arquivos de entrada. Todos estes fatores devem ser considerados para calcular o consumo de energia de uma tarefa. A Equação 4.2 define como é feito o cálculo do consumo total de uma tarefa (FARIA; DANTAS;

CAPRETZ, 2014):

$$C = (E_c \times A) + (E_c \times E) + S + X \quad (4.2)$$

onde E_c é o custo do transporte de um *bit* sobre a rede (BALIGA et al., 2011), A é o tamanho dos arquivos de entrada em *bits*, E é o tamanho do executável em *bit*, S é o custo para alterar o estado do recurso (ligar se estava desligado ou despertar se estava hibernando), X se refere ao consumo de energia no *host* de execução em que a tarefa é executada.

4.5 ALGORITMO ENA

A fim de selecionar o recurso economicamente mais viável, é proposto o algoritmo de seleção de recursos consciente de energia ENA (*Energy and Network Aware algorithm*) (FARIA; DANTAS; CAPRETZ, 2014). O algoritmo proposto é baseado na estratégia WorkQueue (SILVA; CIRNE; BRASILEIRO, 2003). Ao contrário da estratégia usual o algoritmo utiliza informações sobre o ambiente, como o consumo de energia de recursos e estado do nó de execução. O Algoritmo 1 descreve a ideia geral da proposta.

No primeiro passo é criada uma lista de recursos que atendam aos requisitos de cada tarefa, que dependem do tipo de sistema utilizado podendo ser poder computacional, memória disponível, armazenamento em disco, entre outros. O algoritmo leva em consideração recursos em variados estados: desligado, hibernando e ocioso. Caso sejam encontrados recursos que satisfazem os requisitos da tarefa, os nós executores são submetidos ao modelo MEE, com a finalidade de identificar o recurso menos custoso.

Considera-se que o recurso menos custoso é aquele que possui um menor custo total. Para o cálculo do custo energético são considerados: custo de transferência dos arquivos de entrada, custo de execução da tarefa no nó de execução e custo de alteração de estado do nó executor. O custo de alteração de estado se refere ao custo associado à mudança de estado de uma máquina (neste trabalho são considerados estados de hibernação e desligado e o custo associado a ações para modificar o estado do *host* de execução).

Caso o nó não esteja pronto para a execução, uma solicitação é enviada ao nó de execução de acordo com seu estado (hibernando ou desligado), podendo esta ser uma ação de despertar ou ligar o nó.

Algoritmo 1: Pseudocódigo algoritmo ENA

```
1 Inicialização;
2 while Existirem tarefas na fila de execução do escalonador
  do
3   | Verifica recursos disponíveis que satisfaçam requisitos da
   | tarefa;
4   | if Caso encontre recurso disponível then
5   |   | Calcula recurso mais econômico;
6   |   | if estado do recurso = desligado then
7   |   |   | envia solicitação para ligar o recurso;
8   |   |   | else
9   |   |   |   | if estado do recurso = hibernando then
10  |   |   |   |   | envia solicitação para despertar o recurso;
11  |   |   |   |   | end
12  |   |   |   | end
13  |   |   |   | end
13  |   |   |   | Aloca recurso mais energeticamente eficiente;
14  |   | end
15  |   | Mantém tarefa na fila do escalonador;
16  | end
17 end
```

Enfim o recurso é alocado para a execução. Caso não existam recursos disponíveis no *pool* de recursos que satisfaçam os requisitos da tarefa, a mesma é enviada para o fim da fila do escalonador. O algoritmo ENA possui complexidade $O(n)$ onde n é a quantidade de tarefas a serem escalonadas.

5 AMBIENTE E RESULTADOS EXPERIMENTAIS

O objetivo deste capítulo é, além de avaliar o mecanismo de cálculo de consumo, analisar também o desempenho e o consumo de energia do algoritmo desenvolvido neste trabalho em relação a outros algoritmos de escalonamento para grades computacionais identificados na literatura. Para tal, utilizaram-se diversas configurações de máquinas visando aproximar-se de um ambiente o mais heterogêneo possível. Todos os cenários de simulação foram construídos sob o *Framework* SimGrid (LEGRAND; MARCHAL; CASANOVA, 2003). Optou-se pelo SimGrid devido à sua compatibilidade com o modelo proposto, onde o poder computacional é representado por FLOPS, obtendo-se a eficiência energética através do cálculo de FLOPS/watt.

Neste trabalho considerou-se que as tarefas possuem um tamanho fixo de 1 TFLOP e são do tipo BoT, ou seja não possuem relação entre si. As tarefas representam execuções que exigem grande poder computacional e com arquivos de entrada de tamanho variado podendo superar 100 GBs em alguns casos. Estas características de tamanho de arquivos e tarefas se aplicam por exemplo a ambientes de simulação de perfuração de poços de petróleo (MOROOKA et al., 2003), e simulação de variação de modelos climáticos (SOUZA, 2009).

Nas seções que seguem, são apresentados os experimentos preliminares realizados, em seguida são apresentados o ambiente de simulação e os cenários de testes completos, e por fim a análise dos resultados obtidos.

5.1 EXPERIMENTOS PRELIMINARES

Inicialmente o algoritmo foi testado em um ambiente de grade oportunista simulado, onde foi comparado com o algoritmo de seleção de recursos do HTCondor (HTCONDOR, 2014). HTCondor é um sistema especializado em balanceamento de carga voltado a tarefas com processamento intensivo. Atualmente, seu processo de descoberta e seleção de recursos é baseado em um *matching* tradicional, isto é utiliza uma estratégia de seleção similar ao WQ. O objetivo deste experimento preliminar foi comparar o desempenho do algoritmo de seleção de recursos proposto com o do HTCondor em um ambiente de grade oportunista com relação ao consumo de potência necessária para realização de tarefas. A topologia da rede é ilustrada na Figura 15.

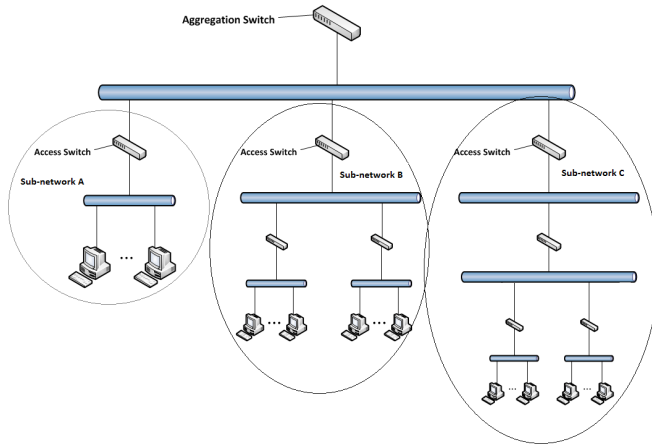


Figura 15: Topologia da rede

A Figura 15 apresenta uma topologia de rede genérica onde pode ser vista uma agregação de *switches* em cascata. Ela representa uma rede de pequeno porte onde máquinas comuns fazem parte de uma grade, interligadas por uma rede de interconexão. As máquinas disponibilizam seus recursos quando estão ociosas para a grade, o que caracteriza um ambiente de grade oportunista. A Tabela 2 exibe a descrição e características coletadas de suas respectivas fichas técnicas dos dispositivos de rede. Os modelos foram selecionados a partir do trabalho proposto por (MONTEIRO; DANTAS; RODRIGUEZ, 2014) e foram considerados dispositivos em pico de trabalho.

Equipamentos		Descrição	Capacidade	Potência (Watts)	
Switches	Acesso	HP/ 3COM SWITCH 4210G	128 GB/s	370	
	Agregação	HP/ 3COM SWITCH 6600	176 GB/s	261	
PC - Desktop HP		Configuração Interna		Iniciar	Despertar
		Core 2 Duo E6850 3,00 GHz 2Gb DDR2/667 MHz		100	85

Tabela 2: Descrição de equipamentos experimentos preliminares

Os testes utilizaram estados aleatórios para as máquinas presentes no ambiente simulado. Os casos de teste gerados consideraram sempre a existência de pelo menos um nó em estado de hibernação. Para analisar a eficácia do algoritmo, dois conjuntos de testes foram realizados. O primeiro usou a configuração padrão HTCCondor, onde cinco testes foram realizados, considerando arquivos de entrada de tamanho

10 MB, 100 MB, 1 GB, 10 GB e 100 GB. Cada teste foi executado cinco vezes para analisar a estratégia de alocação de ambos os cenários. A Figura 16 mostra a alocação de recursos ao fim de seleção e apresenta em números a ordem em que os recursos foram selecionados. A Tabela 3 mostra a potência necessária em watts para executar cada tarefa em cada execução.

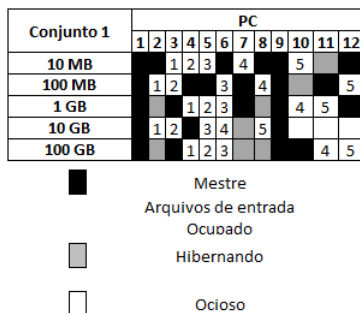


Figura 16: Recursos alocados por HTCCondor

Arquivo de entrada	Potência (Watts)					Potência total (Watts)
	1	2	3	4	5	
10 MB	64,9135	64,91	65,17	64,26	65,17	324,43
100 MB	66,601	73,13	75,73	73,13	75,73	364,34
1 GB	155,35	181,36	181,36	181,36	155,35	854,78
10 GB	324,1	977,5	1237,6	1237,6	977,5	4754,3
100 GB	9199	11800	11800	9199	11800	53798

Tabela 3: Consumo de energia dos recursos alocados pelo sistema HTCCondor

HTCCondor utiliza um algoritmo de seleção de recursos simples, o que implica em selecionar o primeiro recurso disponível. Nós de execução com estado diferente de ocioso não são considerados para a alocação de recursos. Assim sendo, mesmo quando um recurso menos custoso está disponível, ele não será considerado no processo de seleção a menos que se encontre em estado ocioso.

No segundo conjunto de testes, foi utilizado o algoritmo ENA. A Figura 17 mostra a alocação de recursos ao fim de seleção. A figura apresenta em números a ordem em que os recursos foram alocados. A Tabela 4 mostra a potência necessária em watts para executar cada tarefa.

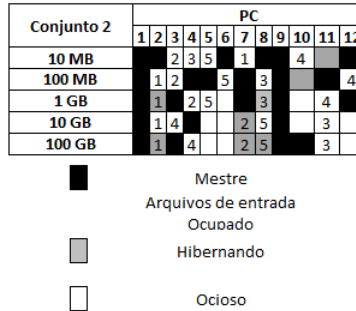


Figura 17: Recursos alocados por ENA

Arquivo de entrada	Potência (Watts)					Potência total (Watts)
	1	2	3	4	5	
10 MB	64,26	64,91	64,91	65,17	65,17	324,43
100 MB	66,60	73,13	73,13	75,73	75,73	364,34
1 GB	111,01	155,35	176,35	155,35	181,36	779,42
10 GB	324,1	345,1	977,5	977,5	977,5	3601,7
100 GB	2686	2686	9199	9199	9220	32990

Tabela 4: Consumo de recursos alocados por ENA

Conforme pode ser observado na Figura 17, o algoritmo ENA optou por utilizar nós em estado de hibernação, o que acabou por ser a abordagem mais eficiente em termos de custo. É também evidente que, para este conjunto de testes durante o teste 3, o algoritmo optou duas vezes por despertar nós (nós 2 e 8) em vez de enviar a tarefa para um nó mais distante. Isso ocorreu porque despertar os nós era menos custoso do que enviar as tarefas para outra sub-rede, considerando que o arquivo de entrada foi de 1 GB.

À medida que o tamanho dos arquivos de entrada aumentou, os testes demonstraram o quanto tornou-se custosa a transferência de arquivos através da rede. Testes de 4 e 5 mostraram a eficiência de se considerar a transferência de arquivos de entrada e estado de nós no processo de seleção. Comparando os resultados, os testes 1 e 2 apresentaram a mesma potência requerida em ambas as abordagens. Quanto aos testes 3, 4 e 5, o algoritmo proposto obteve 8,82%, 24,24% e 38,68% de redução na potência requerida, respectivamente. Os valores totais de potência são mostrados na Figura 18.

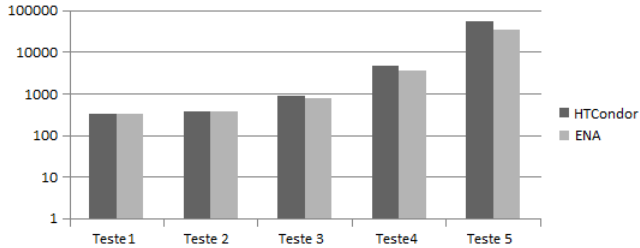


Figura 18: Comparação entre HTCondor e ENA

Ao analisar os valores de ambos os testes, foi determinado que, no pior cenário, o algoritmo ENA obteve o mesmo desempenho que o HTCondor. À medida que o tamanho dos arquivos de entrada aumentou, o desempenho do algoritmo proposto tornou-se superior à estratégia de alocação do HTCondor. Durante os testes 3, 4 e 5, o HTCondor desconsiderou as máquinas próximas ao nó 9 (o nó de armazenamento de arquivos de entrada), devido ao seu estado, embora fossem menos custosos de se utilizar do que máquinas mais distantes ao nó 9. O algoritmo ENA obteve um melhor resultado justamente por considerar máquinas em diferentes estados, como o nó 2, por exemplo, que estava hibernando em testes 3 e 5 antes da seleção e foi despertado. Comparando à potência necessária para executar as tarefas em ambos os cenários, o algoritmo proposto alcançou uma redução total de 36,67% em comparação com o algoritmo de escalonamento do HTCondor.

5.2 AMBIENTE DE SIMULAÇÃO

Nestes experimentos, utilizamos os algoritmos WQ, WQR, Sufferage, XSufferage e DFPLTF, uma vez que estes são algoritmos de escalonamento especializados para grades. A topologia da rede está ilustrada na Figura 19. Apresenta-se um ambiente de nuvem privada, onde o provedor tem acesso completo aos recursos em toda a rede. A topologia baseia-se na rede da Universidade Federal de Santa Catarina (UFSC) (SETIC, 2014). A Tabela 5 apresenta a descrição e características coletadas de suas respectivas fichas técnicas dos dispositivos de rede e computadores pessoais (PCs).

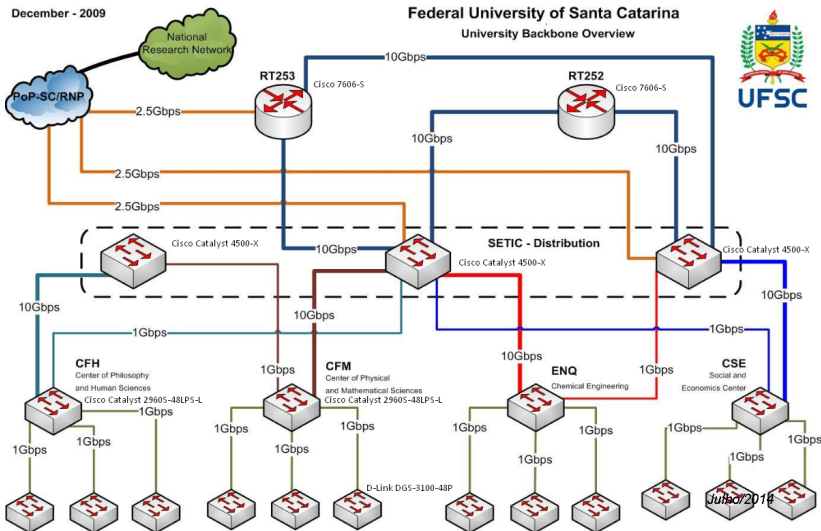


Figura 19: Topologia da rede UFSC

Equipamento	Descrição	Capacidade	Potência (Watts)
Router	RT253/RT252: Cisco 7606-S	480 Gb/s	2700
Access Switch	Cisco Catalyst 4500-X	800 Gb/s	330
Agregation Switch	Cisco Catalyst 2960S-48LPS-L	176 Gb/s	466
Link	D-Link DGS-3100-48P	116 Gb/s	102,6

Tabela 5: Descrição de equipamentos

Encontrar o poder computacional em FLOPS e de eficiência energética em FLOPS/watt de máquinas *off-the-shelf* não é uma tarefa trivial. Por essa razão, os valores de *hosts* em eficiência energética para a simulação foram extraídos das listas Green500 (GREEN500, 2014) e Top500 (TOP500, 2014) (listas de junho de 2014). A Tabela 6 mostra todos os valores utilizados para calcular a eficiência energética de cada *host* simulada de acordo com o seu poder computacional. As máquinas foram escolhidas de acordo com as suas características, tentando manter o ambiente simulado o mais realista e heterogêneo possível.

A classificação das máquinas na lista de Green500 podem ser observadas na segunda coluna da tabela, seguida de seu poder computacional e valores de eficiência energética. Nesta pesquisa, foram utilizados 12 *clusters* espalhados aleatoriamente pela rede. No contexto da simulação, isso se converte em 13 *hosts*, como infraestrutura de uma nuvem privada. Foi considerado um 13º *cluster*, reservado para arqui-

Cluster	Posição Green500	Desempenho (Teraflops)	Eficiência Energética (Megaflops/watt)	Potência Total (KW)	Potência para iniciar (KW)	Custo de execução (W)
12	49	33862,7	1901,54	17808,00	3561,6	525,88
8	16	254,3	2495,12	101,93	20386	400,78
2	2	239,9	3631,7	52,62	10,524	275,35
4	4	153,6	3459,46	44,4	8,88	289,06
11	30	894,4	2176,78	410,9	82,18	459,39
6	10	146,2	2678,41	54,6	10,92	373,35
7	15	709,7	2629,1	269,94	53,988	380,35
5	5	6271	3185,91	1753,66	350,732	313,88
9	17	532,6	2351,1	179,15	35,83	425,33
10	25	715,6	2181,56	328	65,6	458,38
3	3	277,1	3517,84	78,77	15,754	284,26
1	1	150,4	4389,82	34,58	6,916	227,8

Tabela 6: Configuração das máquinas simuladas

vos de entrada, portanto, não é considerado no processo de alocação de recursos.

5.3 RESULTADOS EXPERIMENTAIS

Nesta seção, são apresentados os resultados obtidos nos diferentes cenários dos testes descritos anteriormente. A finalidade é realizar uma análise a respeito do comportamento dos algoritmos de escalonamento testados nos quesitos consumo de energia e tempo de seleção. Os cenários foram definidos com o objetivo de favorecer algoritmos mais simples e rápidos.

Os testes foram divididos em dois cenários. O objetivo dos testes é analisar o desempenho do ENA (FARIA; DANTAS; CAPRETZ, 2014) quando comparado com outros algoritmos. Em ambos os cenários foram considerados arquivos de entrada com tamanho variado: 10 MB, 100 MB, 1 GB, 10 GB, e 100 GB. A fim de analisar a eficácia do algoritmo ENA foram considerados o tempo de execução do algoritmo bem como a diferença entre os mesmos com relação à estimativa de consumo de energia. Nesta seção é feita uma análise para a variação de arquivos de entradas e ao final feita uma análise geral do desempenho do algoritmo proposto comparado aos algoritmos identificados na literatura.

5.3.1 Cenário 1

Considerando-se um cenário em que todos os *hosts* estão disponíveis para seleção, espera-se que o algoritmo mais simples atinja o melhor desempenho geral, neste caso WQ utiliza uma estratégia de alocação simples baseado em uma estratégia FIFO.

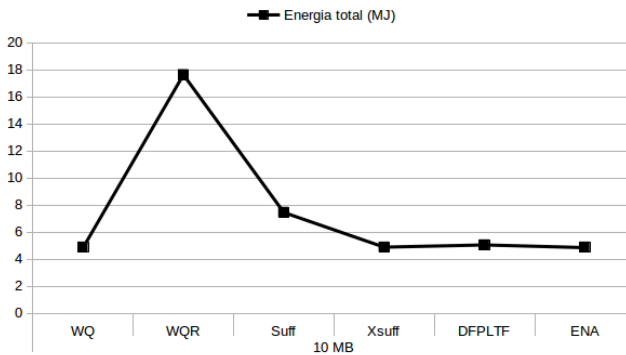


Figura 20: Cenário 1 10MB

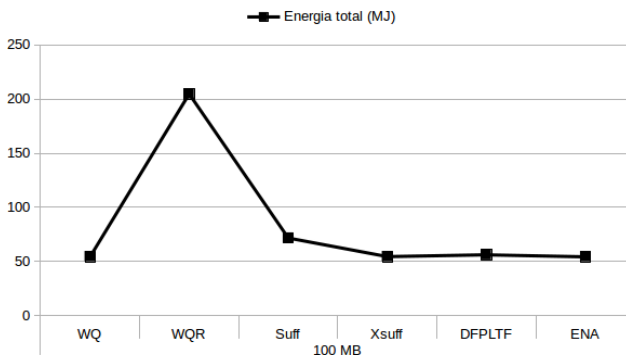


Figura 21: Cenário 1 100MB

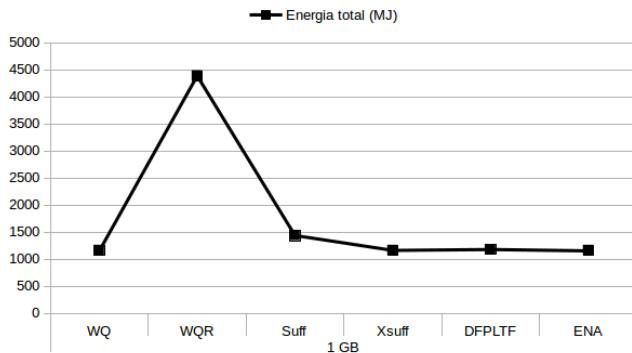


Figura 22: Cenário 1 1GB

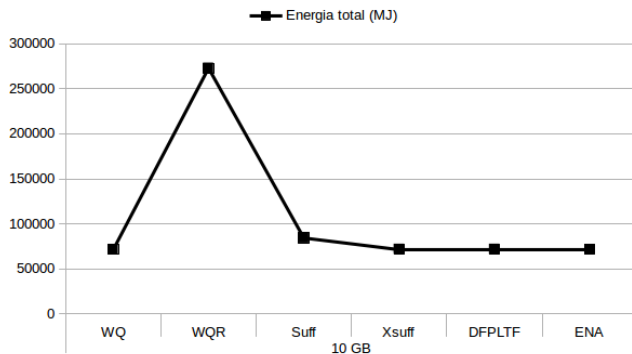


Figura 23: Cenário 1 10GB

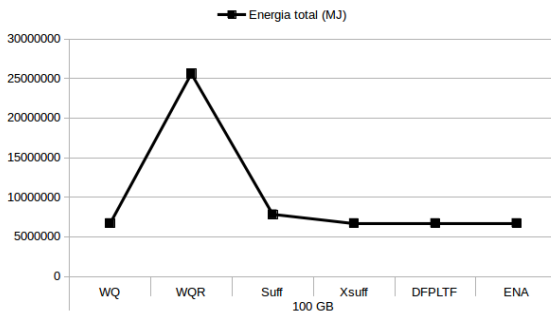


Figura 24: Cenário 1 100GB

As Figuras 20 a 24 apresentam o consumo de energia total dos algoritmos para cada tamanho de arquivo de entrada considerado. Como pode ser observado nas Figuras 20 a 24 existe um padrão para os resultados do Cenário 1, onde o algoritmo WQR obtém os piores resultados, considerando energia consumida. WQ apresentou o melhor resultado como era esperado. Com relação ao ENA, este obteve um resultado semelhante ao WQ, considerando a potência máxima dos dispositivos.

O algoritmo Suff obteve o segundo pior resultado, porém mais próximo dos demais algoritmos, enquanto que Xsuff e DFPLTF se alternaram no terceiro lugar de economia de energia. As Tabelas 7 a 11 apresentam os resultados completos dos testes realizados. Quanto aos tempos, os algoritmos mantiveram um desempenho bastante próximo, havendo apenas uma variação em arquivos pequenos de 10 MB e 100 MB onde os algoritmos Xsuff e DFPLTF demonstraram um ganho.

Arquivo de entrada	Algoritmo	Tempo (s)	Potência (Watts)	Energia (MJ)
10 MB	WQ	1094,69	4471,95	4,89
	WQR	2097,86	8412,38	17,65
	Suff	1253,43	5954,30	7,46
	XSuff	1094,38	4471,95	4,89
	DFPLTF	1094,38	4624,48	5,06
	ENA	1094,69	4471,95	4,89

Tabela 7: Resultado completo Cenário 1 10MB

Arquivo de entrada	Algoritmo	Tempo (s)	Potência (Watts)	Energia (MJ)
100 MB	WQ	10942,20	4995,22	54,66
	WQR	20972,20	9755,95	204,60
	Suff	10941,90	6563,95	71,82
	XSuff	10941,90	4995,22	54,66
	DFPLTF	10941,90	5147,75	56,33
	ENA	10942,20	4995,22	54,66

Tabela 8: Resultado completo Cenário 1 100MB

As Tabelas 7 a 11 apresentam resultados bastante próximos para os algoritmos comparados, exceto o algoritmo WQR que obtém pior resultado tanto em tempo quanto em consumo energético. As tabelas apresentam os algoritmos WQ, XSuff e ENA se alternando como algoritmos com melhor desempenho geral, neste caso o algoritmo mais simples, ou de menor complexidade é o melhor.

Arquivo de entrada	Algoritmo	Tempo (s)	Potência (Watts)	Energia (MJ)
1 GB	WQ	112043,00	10367,41	1161,60
	WQR	214749,00	20438,64	4389,18
	Suff	112043,00	12823,02	1436,73
	XSuff	112043,00	10367,41	1161,59
	DFPLTF	112043,00	10519,94	1178,68
	ENA	112043,00	10367,41	1161,59

Tabela 9: Resultado completo Cenário 1 1GB

Arquivo de entrada	Algoritmo	Tempo (s)	Potência (Watts)	Energia (MJ)
10 GB	WQ	1120430,00	63949,79	71651,26
	WQR	2147480,00	126988,02	272704,22
	Suff	1120430,00	75251,14	84313,63
	XSuff	1120430,00	64102,32	71822,16
	DFPLTF	1120430,00	64102,32	71822,16
	ENA	1120430,00	63949,79	71651,26

Tabela 10: Resultado completo Cenário 1 10GB

Arquivo de entrada	Algoritmo	Tempo (s)	Potência (Watts)	Energia (MJ)
100 GB	WQ	11204300,00	599773,59	6720043,20
	WQR	21474800,00	1192481,79	25608307,84
	Suff	11204300,00	699532,37	7837770,51
	XSuff	11204300,00	599773,59	6720043,20
	DFPLTF	11204300,00	599926,12	6721752,19
	ENA	11204300,00	599773,59	6720043,20

Tabela 11: Resultado completo Cenário 1 100GB

O primeiro cenário considera doze tarefas a serem alocadas em doze *hosts*. A Figura 25 mostra a diferença de energia consumida, entre os resultados do Cenário 1 de cada algoritmo quando comparado com o ENA em porcentagem.

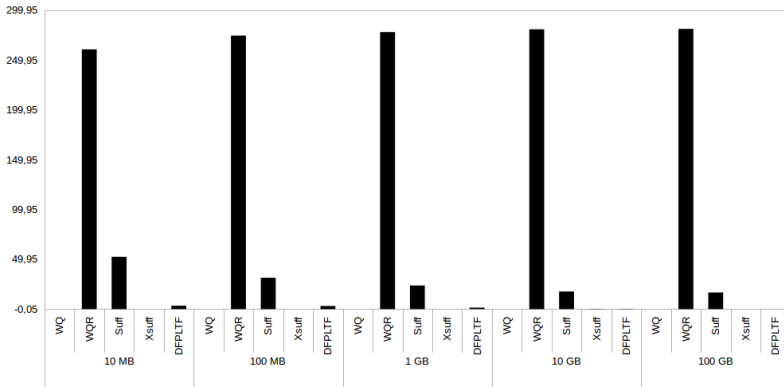


Figura 25: Diferença percentual de energia consumida entre ENA e demais algoritmos para Cenário 1

A Figura 25 mostra que XSuff alcança, no primeiro caso (com 10 MB de arquivos de entrada), o melhor desempenho. A estratégia WQR em todos os cenários testados é a estratégia menos eficiente em termos de consumo de energia. Comparando-se a todos os outros casos, o algoritmo ENA apresenta nos piores casos um desempenho semelhante aos algoritmos WQ, Xsuff e DFPLTF e nos melhores casos apresenta desempenho superior ao Suff e WQR, o único caso onde um algoritmo atinge desempenho superior ao ENA é no caso 1 (com 10 MB de arquivos de entrada), onde o Xsuff supera o ENA. Outra observação importante é que a medida que o tamanho dos arquivos cresce, nota-se uma melhora no desempenho dos algoritmos Suff e DFPLTF.

5.3.2 Cenário 2

No segundo cenário, os três *clusters* mais eficientes em termos de energia foram desligados. Os objetivos do segundo cenário são de comparar a eficiência de estratégias quando existem opções de escolha e minimizar o impacto de *hosts* mais eficientes no processo de seleção. Foram alocadas 6 tarefas para nove *clusters* da grade. Neste caso é esperado que o algoritmo ENA, que considera recursos em estados variados, obtenha os melhores resultados.

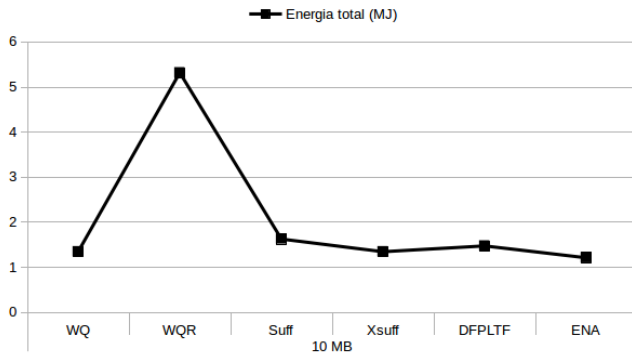


Figura 26: Cenário 2 10MB

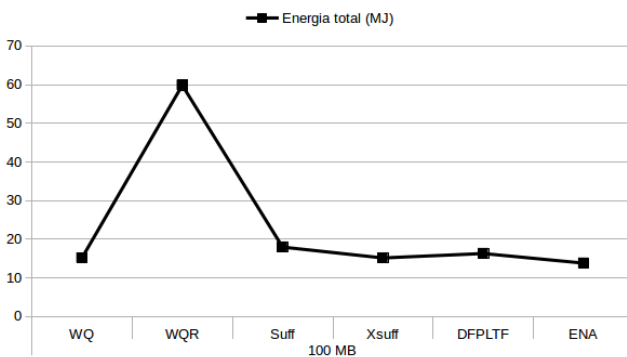


Figura 27: Cenário 2 100MB

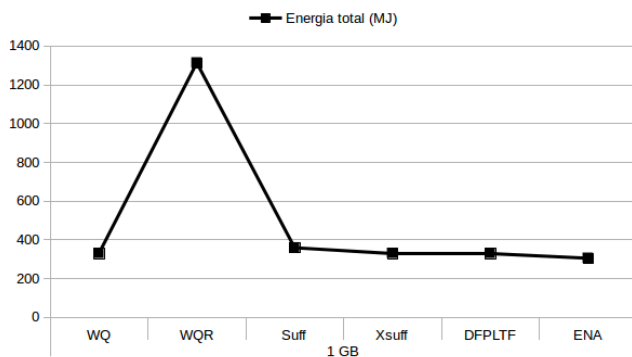


Figura 28: Cenário 2 1GB

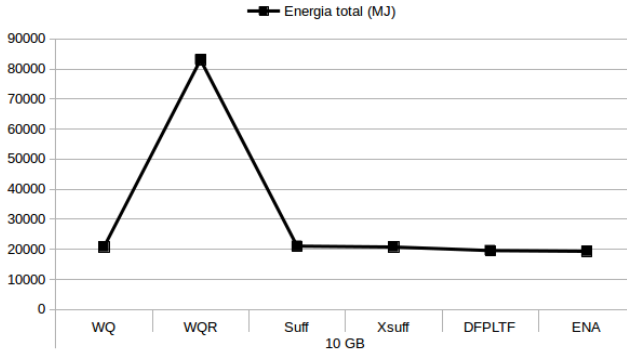


Figura 29: Cenário 2 10GB

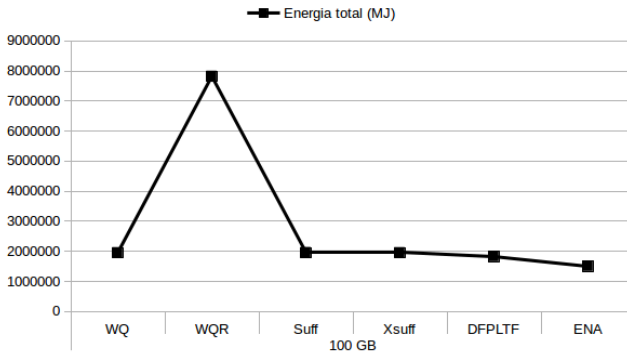


Figura 30: Cenário 2 100GB

Conforme observamos nas Figuras 26 a 30 o padrão observado no Cenário 1 persiste. Porém, o algoritmo ENA obtém os melhores resultados de consumo energético. O algoritmo WQR ainda mantém os piores resultados em tempo de alocação e consumo energético em função de sua estratégia de replicação de tarefas.

O algoritmo Suff manteve o segundo pior resultado, porém mais próximo dos demais algoritmos. Por outro lado, os algoritmos Xsuff e DFPLTF se alternaram no terceiro lugar de economia de energia, a medida que o tamanho dos arquivos de entrada cresceu o desempenho do algoritmo DFPLTF superou o do XSuff. Para arquivos acima de 1 GB o desempenho do algoritmo WQ piorou alternando de posição com XSuff e DFPLTF. Quanto aos tempos, os algoritmos Suff, Xsuff e DFPLTF demonstraram um ganho pequeno, bastante próximo dos

resultados de WQ e ENA. As Tabelas 7 a 11 apresentam os resultados completos dos testes realizados.

Arquivo de entrada	Algoritmo	Tempo (s)	Potência (Watts)	Energia (MJ)
10 MB	WQ	547,48	2462,68	1,35
	WQR	1094,68	4858,89	5,32
	Suff	547,23	2977,15	1,63
	XSuff	547,23	2462,68	1,35
	DFPLTF	547,23	2697,63	1,48
	ENA	547,51	2216,62	1,21

Tabela 12: Resultado completo Cenário 2 10MB

Arquivo de entrada	Algoritmo	Tempo (s)	Potência (Watts)	Energia (MJ)
100 MB	WQ	5471,22	2767,50	15,14
	WQR	10942,20	5468,54	59,84
	Suff	5470,98	3281,97	17,95
	XSuff	5470,98	2767,50	15,14
	DFPLTF	5470,98	2979,35	16,30
	ENA	5471,26	2521,44	13,79

Tabela 13: Resultado completo Cenário 2 100MB

Arquivo de entrada	Algoritmo	Tempo (s)	Potência (Watts)	Energia (MJ)
1 GB	WQ	56021,70	5897,00	330,36
	WQR	112043,00	11727,61	1314,00
	Suff	56021,50	6411,47	359,18
	XSuff	56021,50	5897,00	330,36
	DFPLTF	56021,50	5895,38	330,27
	ENA	56021,70	5447,43	305,17

Tabela 14: Resultado completo Cenário 2 1GB

Arquivo de entrada	Algoritmo	Tempo (s)	Potência (Watts)	Energia (MJ)
10 GB	WQ	560214,00	37110,67	20789,92
	WQR	1120430,00	74155,73	83086,30
	Suff	560213,00	37625,14	21078,09
	XSuff	560213,00	37110,67	20789,88
	DFPLTF	560213,00	34979,95	19596,227
	ENA	560214,00	34532,13	19345,38

Tabela 15: Resultado completo Cenário 2 10GB

Arquivo de entrada	Algoritmo	Tempo (s)	Potência (Watts)	Energia (MJ)
100 GB	WQ	5602130,00	349247,42	1956529,44
	WQR	11204300,00	698436,96	7825497,21
	Suff	5602130,00	349761,89	1959411,57
	XSuff	5602130,00	349247,42	1956529,44
	DFPLTF	5602130,00	325825,69	1825317,87
	ENA	5602130,00	267916,97	1500905,72

Tabela 16: Resultado completo Cenário 2 100GB

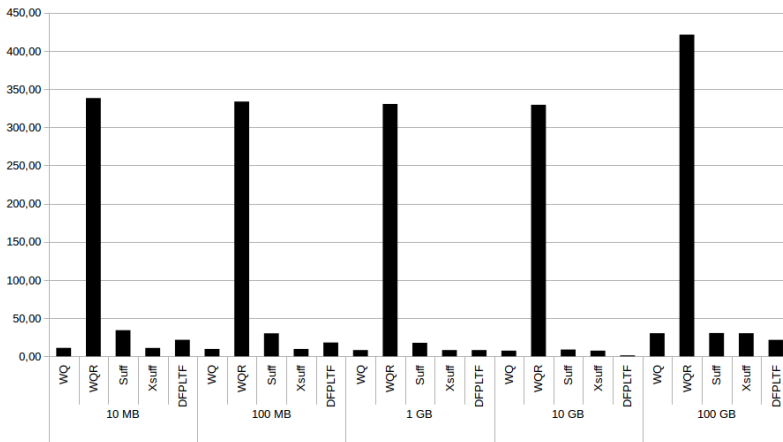


Figura 31: Diferença percentual de energia consumida entre ENA e demais algoritmos para Cenário 2

As Tabelas 7 a 10 apresentam que apesar de o algoritmo ENA levar mais tempo para executar as tarefas, por gastar menos potência ele atinge um consumo de energia inferior aos demais algoritmos.

No segundo cenário o algoritmo ENA conseguiu alcançar o melhor desempenho em todos os casos com relação ao consumo energético total em detrimento do tempo de execução das tarefas, essa abordagem é válida para um conjunto de tarefas que permita esse atraso. A principal razão se dá pelo fato de que o algoritmo ENA considerar *clusters* em estado desligado como uma opção viável para clusters que estão muito longe dos arquivos de entrada. Considerar a variação de estado de *hosts* pode ser considerada uma maneira muito eficaz para economizar energia no ambiente testado.

No melhor dos casos o algoritmo ENA atingiu um ganho próximo de 420%, quando comparado com a estratégia WQR, a pior das estratégias em termos de eficiência energética. Quando comparado com a estratégia WQ, a estratégia mais simples, ENA conseguiu ser melhor em todos os casos. A maior diferença pode ser vista quando o arquivo de entrada é o maior (100 GB) em torno de 30%. O principal motivo é devido ao ENA considerar os *hosts* em estado desligado além do custo para transferir arquivos pela rede, optando por *hosts* mais próximos dos arquivos de entrada, quando estes aumentam muito o tamanho. Nos demais casos estudados, em especial no segundo conjunto de testes, o ENA obtém um melhor desempenho energético em detrimento do relaxamento do tempo de computação das tarefas.

6 CONCLUSÕES E TRABALHOS FUTUROS

Este trabalho apresentou uma revisão de literatura sobre arquiteturas computacionais paralelas e distribuídas, abordando conceitos e características que os tornam ambientes complexos. Devido à complexidade destes ambientes, foi evidenciada a importância de estudos na área de escalonamento de tarefas com o objetivo de encontrar formas para reduzir seu consumo de energia. Além disso, foram apresentadas políticas de escalonamento como WQ, WQR, *Sufferage*, *XSufferage* e *Dynamic FPLTF*. O estudo de abordagens utilizadas na literatura apontou para uma tendência em se considerar variação de estado de máquinas ociosas como principal abordagem para redução de consumo energético em ambientes de alto desempenho.

Neste contexto, foi proposta uma abordagem de seleção de recursos consciente de energia. A abordagem envolveu a proposta de um modelo de cálculo de consumo e um algoritmo de escalonamento de recursos denominado ENA (*Energy and Network Aware algorithm*), a fim de validar o modelo. Ele realiza a seleção com base no custo associado com a transferência de dados através da rede, de mudança de estado de recursos e de execução de tarefas. O principal objetivo da abordagem foi reduzir o custo associado à execução de aplicações submetidas a um *pool* de recursos através da seleção do recurso disponível menos custoso e, conseqüentemente, reduzir as despesas de energia da infraestrutura.

Uma característica desejada da abordagem seria que esta fosse genérica, no sentido de se poder realizar alocação de recursos em qualquer tipo de arquitetura paralela ou distribuída. Por este motivo, para validar a proposta foi escolhida a arquitetura de uma nuvem computacional, que pode ter em sua infraestrutura *clusters*, *multi-clusters*, computadores pessoais e ainda grades computacionais, desde que seu gerenciador de recurso tenha domínio sob os mesmos.

A proposta foi avaliada através de simulações realizadas no simulador de grade SimGrid, associado à biblioteca LIBTS. O ambiente simulado representa uma infraestrutura de nuvem computacional formada por *clusters* dispersos geograficamente e interconectados. Os resultados mostraram que a abordagem proposta, na maioria dos casos, escolhe o melhor recurso disponível e no pior dos casos tem um desempenho igual ao algoritmo mais simples encontrado na literatura, o algoritmo WQ. O algoritmo ENA, alcançou resultados bastante expressivos quando comparado com algoritmos com pior desempenho. Quando considerados os algoritmos dinâmicos XSuff e DFPLTF o al-

goritmo ENA obteve um melhor resultado a medida que o tamanho de arquivos de entrada aumentaram.

Em termos de trabalhos futuros, espera-se realizar os seguintes experimentos e extensões da abordagem proposta:

- Estudar a adição de uma estratégia de migração de tarefas com base no estado da rede e de nós de execução, utilizando-se por exemplo *Software Defined Networks*(SDN). SDNs permitem a administradores de rede definir rotas para transferência de dados, o que associado ao modelo proposto pode gerar rotas de SDN conscientes de consumo energético;
- Aplicar a abordagem de seleção de recursos proposta em ambientes de nuvem real, criando-se um ambiente de nuvem privada consciente de consumo energético;
- Considerando a abordagem como um primeiro passo para uma caracterização de tarefas, também é uma possibilidade a criação de perfis para as tarefas e realizar a seleção de recursos com base em histórico de alocação;
- Estudar a viabilidade de mecanismos de reserva antecipada de recursos conscientes de energia. Utilizando-se o modelo de consumo energético seria possível realizar uma reserva antecipada de recurso baseado no consumo energético dos recursos, mesmo que estes não estejam disponíveis no momento da seleção. Neste contexto, pode-se ainda estudar *trade-offs* de reserva antecipada de recursos conscientes de energia e desempenho de sistemas;
- Aplicar o modelo de seleção de recursos a outros ambientes como, por exemplo, uma federação de nuvens, onde, considerando a necessidade de mais recursos para uma nuvem é necessário realizar um empréstimo de recursos de outras nuvens pertencentes a federação. A ideia envolve aplicar o modelo de consumo energético proposto no processo de seleção de recursos externos para avaliar o recurso menos custoso.
- Elaborar um modelo de alocação de tarefas para sistemas distribuídos de tempo real, que considera o consumo de energia conforme proposto neste trabalho.

REFERÊNCIAS

- ABAWAJY, J. H.; DANDAMUDI, S. P. Parallel job scheduling on multicluster computing system. In: **IEEE. Cluster Computing, 2003. Proceedings. 2003 IEEE International Conference on.** [S.l.], 2003. p. 11–18.
- ARMBRUST, M. et al. M.: Above the clouds: A berkeley view of cloud computing. Citeseer, 2009.
- BALIGA, J. et al. Green cloud computing: Balancing energy in processing, storage, and transport. **Proceedings of the IEEE, IEEE**, v. 99, n. 1, p. 149–167, 2011.
- BRASILEIRO, F. et al. An approach for the co-existence of service and opportunistic grids: The eela-2 case. In: **Latin-American Grid Workshop.** [S.l.: s.n.], 2008.
- BUYYA, R. (Ed.). **High Performance Cluster Computing: Architectures and Systems.** Prentice hall. [S.l.: s.n.], 1999.
- BUYYA, R.; MURSHED, M. Gridsim: A toolkit for the modeling and simulation of distributed resource management and scheduling for grid computing. **Concurrency and computation: practice and experience**, Wiley Online Library, v. 14, n. 13-15, p. 1175–1220, 2002.
- BUYYA, R.; SULISTIO, A. Service and utility oriented distributed computing systems: challenges and opportunities for modeling and simulation communities. In: **IEEE. Simulation Symposium, 2008. ANSS 2008. 41st Annual.** [S.l.], 2008. p. 68–81.
- CAMARGO, R. Y. D. et al. The grid architectural pattern: Leveraging distributed processing capabilities. **Pattern Languages of Program Design**, v. 5, p. 337–356, 2006.
- CAPPELLO, F. et al. Grid'5000: A large scale and highly reconfigurable grid experimental testbed. In: **IEEE COMPUTER SOCIETY. Proceedings of the 6th IEEE/ACM International Workshop on Grid Computing.** [S.l.], 2005. p. 99–106.
- CARISSIMI, A. D. S. **Athapascan-0: Exploitation de la multiprogrammation légère sur grappes de multiprocesseurs.** Tese (Doutorado), 1999.

CASAVANT, T. L.; KUHL, J. G. A taxonomy of scheduling in general-purpose distributed computing systems. **Software Engineering, IEEE Transactions on**, IEEE, v. 14, n. 2, p. 141–154, 1988.

CASTRO, H. et al. Green flexible opportunistic computing with virtualization. In: IEEE. **Computer and Information Technology (CIT), 2011 IEEE 11th International Conference on**. [S.l.], 2011. p. 629–634.

CASTRO, M. et al. Energy efficient seismic wave propagation simulation on a low-power manycore processor. In: IEEE. **Computer Architecture and High Performance Computing (SBAC-PAD), 2014 IEEE 26th International Symposium on**. [S.l.], 2014. p. 57–64.

COULOURIS, G.; DOLLIMORE, J.; KINDBERG, T. **Sistemas Distribuídos: Conceitos e Projeto**. [S.l.]: Bookman, 2007.

CRUZ, E. H.; ALVES, M. A.; NAVAU, P. O. Process mapping based on memory access traces. In: IEEE. **Computing Systems (WSCAD-SCC), 2010 11th Symposium on**. [S.l.], 2010. p. 72–79.

DANTAS, M. **Computação distribuída de alto desempenho: redes, clusters e grids computacionais**. [S.l.]: Axcel Books, 2005.

DINECHIN, B. D. de et al. A distributed run-time environment for the kalray mppa®-256 integrated manycore processor. **Procedia Computer Science**, Elsevier, v. 18, p. 1654–1663, 2013.

EC2, A. **Amazon EC2**. Acessado em: Junho 2014. Disponível em: <<http://aws.amazon.com/ec2>>.

FAN, X.; WEBER, W.-D.; BARROSO, L. A. Power provisioning for a warehouse-sized computer. In: ACM. **ACM SIGARCH Computer Architecture News**. [S.l.], 2007. v. 35, n. 2, p. 13–23.

FARIA, I. D. et al. Network and energy-aware resource selection model for opportunistic grids. In: IEEE. **WETICE Conference (WETICE), 2014 IEEE 23rd International**. [S.l.], 2014. p. 167–172.

FARIA, I. de; DANTAS, M.; CAPRETZ, M. A. Energy-aware resource selection on opportunistic grids. In: IEEE. **Computers and**

Communication (ISCC), 2014 IEEE Symposium on. [S.l.], 2014. p. 1–6.

FERNÁNDEZ-MONTES, A. et al. Smart scheduling for saving energy in grid computing. **Expert Systems with Applications**, Elsevier, v. 39, n. 10, p. 9443–9450, 2012.

FERREIRA, D. J. **Reserva Dinamica e Antecipada de Recursos para Configuracoes Multi-Clusters Utilizando Ontologias e Logica Difusa.** Dissertação (Mestrado) — Universidade Federal de Santa Catarina, Agosto 2010.

FLYNN, M. J. Some computer organizations and their effectiveness. **Computers, IEEE Transactions on**, IEEE, v. 100, n. 9, p. 948–960, 1972.

FOSTER, I.; KESSELMAN, C. **The Grid: Blueprint for a New Computing Infrastructure.** [S.l.]: Morgan Kaufmann, 2004.

FOSTER, I.; KESSELMAN, C. The History of the grid. **High Performance Computing: From Grids and Clouds to Exascale, Advances in Parallel Computing Series.**, Vol. 20, p. 3–30, 2011. IOS Press.

FOSTER, I. et al. Cloud computing and grid computing 360-degree compared. In: IEEE. **Grid Computing Environments Workshop, 2008. GCE'08.** [S.l.], 2008. p. 1–10.

FRANCO, P. B. Escalonamento de tarefas em ambiente de simulação de grid computacional. Universidade Estadual Paulista (UNESP), 2011.

GARFINKEL, S. L. **Architects of the Information Society: Thirty-Five Years of the Laboratory for Computer Science at MIT.** [S.l.]: The MIT Press, 1999.

GARG, S. K.; BUYYA, R. Exploiting heterogeneity in grid computing for energy-efficient resource allocation. In: **Proceedings of the 17th International Conference on Advanced Computing and Communications (ADCOM 2009), Bengaluru, India.** [S.l.: s.n.], 2009.

GREEN500. **The Green 500 List.** Acessado em: Junho 2014. Disponível em: <<http://www.green500.org/>>.

- GREENBERG, A. et al. The cost of a cloud: research problems in data center networks. **ACM SIGCOMM computer communication review**, ACM, v. 39, n. 1, p. 68–73, 2008.
- GUENTER, B.; JAIN, N.; WILLIAMS, C. Managing cost, performance, and reliability tradeoffs for energy-aware server provisioning. In: IEEE. **INFOCOM, 2011 Proceedings IEEE**. [S.l.], 2011. p. 1332–1340.
- HAMILTON, J. Cooperative expendable micro-slice servers (cems): low cost, low power servers for internet-scale services. In: **Conference on Innovative Data Systems Research**. [S.l.: s.n.], 2009.
- HERMENIER, F. et al. Entropy: a consolidation manager for clusters. In: ACM. **Proceedings of the 2009 ACM SIGPLAN/SIGOPS international conference on Virtual execution environments**. [S.l.], 2009. p. 41–50.
- HEWITT, C. Orgs for scalable, robust, privacy-friendly client cloud computing. **Internet Computing, IEEE**, IEEE, v. 12, n. 5, p. 96–99, 2008.
- HTCONDOR. **HTCondor**. Acessado em: Junho 2014. Disponível em: <<http://research.cs.wisc.edu/htcondor/>>.
- HUEDO, E.; MONTERO, R. S.; LLORENTE, I. M. **An experimental framework for executing applications in dynamic Grid environments**. [S.l.], 2002.
- JAVADI, B.; AKBARI, M. K.; ABAWAJY, J. H. A performance model for analysis of heterogeneous multi-cluster systems. **Parallel computing**, Elsevier, v. 32, n. 11, p. 831–851, 2006.
- KIM, K. H.; BUYYA, R.; KIM, J. Power aware scheduling of bag-of-tasks applications with deadline constraints on dvs-enabled clusters. In: **CCGRID**. [S.l.: s.n.], 2007. v. 7, p. 541–548.
- KON, F.; GOLDMAN, A. Grades Computacionais: Conceitos Fundamentais e Casos Concretos. **Proceedings of the Tomasz Kowaltowski e Karin Breitman (Org.) Atualizações em Informática**, p. 55–104, 2008.
- KOOMEY, J. G. Estimating regional power consumption by servers: A technical note. **Lawrence Berkeley National Laboratory, Oakland, CA**, 2007.

KRAUTER, K.; BUYYA, R.; MAHESWARAN, M. A taxonomy and survey of grid resource management systems for distributed computing. **Software: Practice and Experience**, Wiley Online Library, v. 32, n. 2, p. 135–164, 2002.

LASZEWSKI, G. V. et al. Power-aware scheduling of virtual machines in dvfs-enabled clusters. In: IEEE. **Cluster Computing and Workshops, 2009. CLUSTER'09. IEEE International Conference on**. [S.l.], 2009. p. 1–10.

LEGRAND, A.; MARCHAL, L.; CASANOVA, H. Scheduling distributed applications: the simgrid simulation framework. In: IEEE. **Cluster Computing and the Grid, 2003. Proceedings. CCGrid 2003. 3rd IEEE/ACM International Symposium on**. [S.l.], 2003. p. 138–145.

LIU, H. et al. Performance and energy modeling for live migration of virtual machines. **Cluster computing**, Springer, v. 16, n. 2, p. 249–264, 2013.

LIU, Y.; ZHU, H. A survey of the research on power management techniques for high-performance systems. **Software: Practice and Experience**, Wiley Online Library, v. 40, n. 11, p. 943–964, 2010.

MALONE, C.; BELADY, C. Metrics to characterize data center & it equipment energy use. In: **Proceedings of the Digital Power Forum, Richardson, TX**. [S.l.: s.n.], 2006.

MÄMMELÄ, O. et al. Energy-aware job scheduler for high-performance computing. **Computer Science-Research and Development**, Springer, v. 27, n. 4, p. 265–275, 2012.

MEFFE, C.; MUSSI, E.; MELLO, L. Guia de Estruturação e Administração do Ambiente de Cluster e Grid. **Brasilia**, 2006.

MEI, C. et al. Optimizing a parallel runtime system for multicore clusters: a case study. In: ACM. **Proceedings of the 2010 TeraGrid Conference**. [S.l.], 2010. p. 12.

MELL, P.; GRANCE, T. The NIST definition of cloud computing (draft). **NIST special publication**, v. 800, p. 145, 2011.

MENASCÉ, D. A. et al. Static and dynamic processor scheduling disciplines in heterogeneous parallel architectures. **Journal of Parallel and Distributed Computing**, Elsevier, v. 28, n. 1, p. 1–18, 1995.

MONTEIRO, R. C.; DANTAS, M.; RODRIGUEZ, M. V. R. y. Green cloud computing: An experimental validation. In: IOP PUBLISHING. **Journal of Physics: Conference Series**. [S.l.], 2014. v. 540, n. 1, p. 012005.

MONTERO, R. S.; HUEDO, E.; LLORENTE, I. M. Grid resource selection for opportunistic job migration. In: **Euro-Par 2003 Parallel Processing**. [S.l.]: Springer, 2003. p. 366–373.

MORARI, A. et al. Efficient sorting on the tilera manycore architecture. In: IEEE. **Computer Architecture and High Performance Computing (SBAC-PAD), 2012 IEEE 24th International Symposium on**. [S.l.], 2012. p. 171–178.

MOROOKA, C. et al. Dynamic behavior of a vertical production riser by quasi 3d calculations. In: AMERICAN SOCIETY OF MECHANICAL ENGINEERS. **ASME 2003 22nd International Conference on Offshore Mechanics and Arctic Engineering**. [S.l.], 2003. p. 293–300.

MURSHED, M.; BUYYA, R. et al. Using the gridsim toolkit for enabling grid computing education. In: **International Conference on Communication Networks and Distributed Systems Modeling and Simulation**. [S.l.: s.n.], 2002. p. 27–31.

NESMACHNOW, S. et al. Energy-aware scheduling on multicore heterogeneous grid computing systems. **Journal of grid computing**, Springer, v. 11, n. 4, p. 653–680, 2013.

NGS. **National Grid Service**. Acessado em: Junho 2014. Disponível em: <<http://www.ngs.ac.uk/>>.

PAMLIN, D. The potential global co2 reductions from ict use: Identifying and assessing the opportunities to reduce the first billion tonnes of co2. **WWF, Sweden (May 2008)**, 2008.

PONCIANO, L.; BRASILEIRO, F. On the impact of energy-saving strategies in opportunistic grids. In: IEEE. **Grid Computing (GRID), 2010 11th IEEE/ACM International Conference on**. [S.l.], 2010. p. 282–289.

QURESHI, A. **Power-Demand Routing in Massive Geo-Distributed Systems**. Tese (Doutorado) — Massachusetts Institute of Technology, 2010.

RAJOVIC, N. et al. The low power architecture approach towards exascale computing. **Journal of Computational Science**, Elsevier, v. 4, n. 6, p. 439–443, 2013.

REIS, V. Q. d. **Escalonamento em grids computacionais: estudo de caso**. Tese (Doutorado) — Universidade de São Paulo, 2005.

RIBEIRO, C. P. et al. Memory affinity for hierarchical shared memory multiprocessors. In: IEEE. **Computer Architecture and High Performance Computing, 2009. SBAC-PAD'09. 21st International Symposium on**. [S.l.], 2009. p. 59–66.

SETIC. **Superintendência de Governança Eletrônica e Tecnologia da Informação e Comunicação**. Acessado em: Junho 2014. Disponível em: <<http://www.setic.ufsc.br/>>.

SiliconValley Leadership Group. **Data Centre Energy Forecast Report**. [S.l.], 2008. Disponível em: <<http://www.accenture.com/SiteCollectionDocuments/PDF/Accenture-Data-Center-Energy-Forecast.pdf>>.

SILVA, D. P. D.; CIRNE, W.; BRASILEIRO, F. V. Trading cycles for information: Using replication to schedule bag-of-tasks applications on computational grids. In: **Euro-Par 2003 Parallel Processing**. [S.l.]: Springer, 2003. p. 169–180.

SINAPAD. **Sistema Nacional de Processamento de Alto Desempenho**. Acessado em: Junho 2014. Disponível em: <<https://www.lncc.br/sinapad/index.php>>.

SOUZA, E. B. D. Modelo olam (ocean-land-atmosphere-model): descrição, aplicações, e perspectivas. **Revista Brasileira de Meteorologia**, SciELO Brasil, v. 24, n. 2, p. 144–157, 2009.

SULISTIO, A.; YEO, C. S.; BUYYA, R. A taxonomy of computer-based simulations and its mapping to parallel and distributed systems simulation tools. **Software: Practice and Experience**, Wiley Online Library, v. 34, n. 7, p. 653–673, 2004.

TANENBAUM, A. S.; STEEN, M. V. **Distributed systems**. [S.l.]: Prentice Hall, 2002.

TANENBAUM, A. S.; WOODHULL, A. S. **Sistemas Operacionais: Projetos e Implementação**. [S.l.]: Bookman, 2006.

TEODORO, S.; CARMO, A. B. do; FERNANDES, L. G. Energy efficiency management in computational grids through energy-aware scheduling. In: **ACM. Proceedings of the 28th Annual ACM Symposium on Applied Computing**. [S.l.], 2013. p. 1163–1168.

TOP500. **Top 500 The List**. Acessado em: Junho 2014. Disponível em: <<http://www.top500.org/>>.

TOTONI, E. et al. Comparing the power and performance of intel's scc to state-of-the-art cpus and gpus. In: IEEE. **Performance Analysis of Systems and Software (ISPASS), 2012 IEEE International Symposium on**. [S.l.], 2012. p. 78–87.

VARGHESE, A. et al. Programming the adapteva epiphany 64-core network-on-chip coprocessor. In: IEEE. **Parallel & Distributed Processing Symposium Workshops (IPDPSW), 2014 IEEE International**. [S.l.], 2014. p. 984–992.

VERMA, A.; AHUJA, P.; NEOGI, A. pMapper: Power and Migration Cost Aware Application Placement in Virtualized Systems. p. 243–264, 2008.

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

ZHU, Q.; ZHU, J.; AGRAWAL, G. Power-aware consolidation of scientific workflows in virtualized environments. In: IEEE COMPUTER SOCIETY. **Proceedings of the 2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis**. [S.l.], 2010. p. 1–12.

APÊNDICE A – Publicações

A.1 TRABALHOS COMPLETOS PUBLICADOS EM ANAIS DE CONGRESSOS

Este apêndice visa ilustrar as publicações realizadas ao longo da pesquisa desse trabalho de dissertação.

1. Energy-aware resource selection on opportunistic grids

- **Autores:** Faria, Izaias de; Dantas, M.A.R. ; Capretz, Miriam A.M ; Higashino, Wilson A
- **Evento:** ISCC 2014, 2014 IEEE Symposium on Computers and Communication (ISCC)
- **Local:** Madeira, Portugal
- **Data:** junho de 2014
- **Estrato Qualis/CAPES:** A2
- **Descrição:** Proposta de um modelo de seleção de recursos consciente de consumo energético que considera: estado de nós de execução, custo de transferência de arquivos e custo de execução de tarefas em nós de execução. A proposta foi validada em um ambiente de grade oportunista simulado. Testes consideraram tarefas com arquivos de entrada variando entre 10 MB e 100 GB em um pequeno conjunto de nós de execução.

2. Network and Energy-Aware Resource Selection Model for Opportunistic Grids

- **Autores:** De Faria, Izaias ; Dantas, M.A.R. ; Capretz, Miriam A.M
- **Evento:** WETICE 2014, IEEE 23rd International Workshops on Enabling Technologies: Infrastructures for Collaborative Enterprise (WETICE)
- **Local:** Parma, Italia
- **Data:** Junho de 2014
- **Estrato Qualis/CAPES:** B1
- **Descrição:** Proposta de um modelo de seleção de recursos consciente de consumo energético que considera: estado de nós de execução, custo de transferência de arquivos, custo de execução de tarefas em nós de execução e latência de rede.

A proposta foi validada em um ambiente de grade oportunista simulado. Testes consideraram tarefas com arquivos de entrada variando entre 10 MB e 100 MB em um grande conjunto de nós de execução.