PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO GRANDE DO SUL
FACULDADE DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Algoritmos de Escalonamento para Grades Computacionais voltados à Eficiência Energética

SILVANA TEODORO

Dissertação apresentada como requisito parcial à obtenção do grau de Mestre em Ciência da Computação na Pontifícia Universidade Católica do Rio Grande do Sul.

Orientador: Prof. Dr. Luiz Gustavo Leão Fernandes

Porto Alegre
2013

Dados Internacionais de Catalogação na Publicação (CIP)

T314a Teodoro, Silvana
Algoritmos de escalonamento para grades computacionais voltados à eficiência energética / Silvana Teodoro. – Porto Alegre, 2013.
117 f.

Diss. (Mestrado) – Fac. de Informática, PUCRS.
Orientador: Prof. Dr. Luiz Gustavo Leão Fernandes.

1. Informática. 2. Processamento de Alto Desempenho. 3. Algoritmos (Programação). 4. Energia Elétrica. – Conservação. I. Fernandes, Luiz Gustavo Leão. II. Título.

CDD 004.22

Ficha Catalográfica elaborada pelo
Setor de Tratamento da Informação da BC-PUCRS

Pontifícia Universidade Católica do Rio Grande do Sul
FACULDADE DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

TERMO DE APRESENTAÇÃO DE DISSERTAÇÃO DE MESTRADO

Dissertação intitulada "Algoritmos de Escalonamento para Grades Computacionais voltados à Eficiência Energética" apresentada por Silvana Teodoro como parte dos requisitos para obtenção do grau de Mestre em Ciência da Computação, Processamento Paralelo e Distribuído, aprovada em 13/03/2013 pela Comissão Examinadora:

Prof. Dr. Luiz Gustavo Leão Fernandes – Orientador — PPGCC/PUCRS

Prof. Dr. Fabiano Passuelo Hessel – PPGCC/PUCRS

Prof. Dr. Tiago Coelho Ferreto – PPGCC/PUCRS

Prof. Dr.Claudio Fernando Resin Geyer – UFRGS

Homologada em 30/04/2013, conforme Ata No. 007 pela Comissão Coordenadora.

Prof. Dr. Paulo Henrique Lemelle Fernandes
Coordenador.

PUCRS
Campus Central
Av. Ipiranga, 6681 – P32– sala 507 – CEP: 90619-900
Fone: (51) 3320-3611 – Fax (51) 3320-3621
E-mail: ppgcc@pucrs.br
www.pucrs.br/facin/pos

AGRADECIMENTOS

Primeiramente agradeço a Deus por ter me permitido chegar até aqui.

À minha família, pelo carinho e apoio nos momentos em que precisei. Em especial, agradeço aos meus pais, meus incentivadores incansáveis. Meus exemplos de vida, de integridade, dignidade e honradez. Sem vocês eu não seria nada. E só vocês sabem o quanto sonhei em chegar até aqui.

Ao meu “namorido” Guilherme, pelo apoio, compreensão e ajuda. Pelos momentos que vivemos durante esses dois grandes anos de Mestrado. Que continuemos sempre juntos para o que der e vier, sobre a sombra do nosso amor.

Aos colegas do Grupo de Modelagem de Aplicações Paralelas (GMAP), Viviane, Andriele, Dalvan, Victoria, Eduardo, Mateus, Carol e Rafael, pela ajuda, apoio e troca de conhecimento e pelas festinhas de aniver, jantas e almoços do grupo.

Aos professores da PUCRS pelo conhecimento transmitido. Em especial agradeço ao Prof. Dr. Luiz Gustavo Leão Fernandes, pela orientação nesta Dissertação. E pela relação humana, transparente e de amizade com que trata seus orientandos. Agradeço também aos funcionários da PPGCC pela dedicação em nos manter sempre informados de toda burocracia e prazos. E à FAPERGS que ofereceu suporte financeiro para o desenvolvimento deste trabalho.

Aos meus amigos, pelos momentos de descontração e pelos encontros impagáveis, que inúmeras vezes recuperaram minhas forças e me deram ânimo novo para continuar.

Meu muito obrigado a todos! Com certeza sem vocês a realização deste trabalho não seria possível. Todos são de alguma forma, parte fundamental da jornada.

ALGORITMOS DE ESCALONAMENTO PARA GRADES COMPUTACIONAIS VOLTADOS À EFICIÊNCIA ENERGÉTICA

RESUMO

Os recentes avanços da Computação de Alto Desempenho abrem um largo espectro de possibilidades para as pesquisas na área. Arquiteturas paralelas e distribuídas modernas apresentam cada vez mais capacidade de processamento em busca de um maior poder computacional. Ao mesmo tempo, o ganho de desempenho obtido com as plataformas é seguido por um aumento do consumo de energia. Neste cenário, pesquisas sobre eficiência energética em ambientes de alto desempenho têm surgido como uma forma de encontrar as causas e propor soluções para o consumo excessivo de energia. Atualmente, uma das mais representativas plataformas de alto desempenho é a grade computacional, que é usada em muitos projetos científicos e acadêmicos em todo mundo. Neste trabalho, propomos o uso de algoritmos de escalonamento de tarefas energeticamente eficientes para a gestão do consumo de energia em grades computacionais sem causar perdas significativas de desempenho. A solução é baseada em: (i) gestão eficiente de recursos ociosos; (ii) uso inteligente de recursos ativos; (iii) desenvolvimento de um mecanismo para estimar com precisão a energia consumida por uma determinada plataforma; (iv) proposta de novos algoritmos de escalonamento energeticamente eficientes para grades computacionais. A abordagem criada foi avaliada utilizando o ambiente de simulação SimGrid. Comparamos nossos algoritmos com cinco algoritmos de escalonamento tradicionais para grades computacionais, que não consideram questões de energia, e um algoritmo recentemente proposto na literatura que lida com questões de consumo de energia. Os resultados mostram, em alguns cenários, uma redução no consumo de energia de 221,03%, combinada com uma perda de desempenho de 34,60%, com o uso de um dos algoritmos desenvolvidos neste trabalho. Este exemplo confirma a nossa hipótese de que é possível reduzir significativamente o consumo de energia em uma grade computacional sem comprometer de forma proporcional o desempenho.

Palavras-chave: Computação de Alto Desempenho, algoritmos de escalonamento, eficiência energética, simulação.

ENERGY-AWARE SCHEDULING ALGORITHMS FOR COMPUTATIONAL GRIDS

ABSTRACT

Recent advances in High Performance Computing have opened a wide range of new research opportunities. Modern parallel and distributed architectures present each time more and more processing units seeking for a higher computational power. At the same time, the gain of performance obtained with those platforms is followed by an increase in energy consumption. In this scenario, researches in energy efficient high performance environments have emerged as a way to find the causes of excessive energy consumption and propose alternative solutions. Nowadays, one of the most representative high performance platforms is the computational grid which is used in many scientific and academic projects all over the world. In this work, we propose the use of energy-aware scheduling algorithms to efficiently manage the energy consumption in computational grids trying to avoid excessive performance losses. Our solution is based on: (i) an efficient management of idle resources; (ii) a clever use of active resources; (iii) the development of a procedure to accurately estimate the energy consumed in a given platform; (iv) the proposal of several new energy-aware scheduling algorithms for computational grids. We evaluate our approach using the SimGrid simulation environment and we compared our algorithms against five traditional scheduling algorithms for computational grids that are not energy-aware and one new algorithm recently proposed in the literature that deals with energy consumption issues. Our results show that in some experimental scenarios using our algorithms it is possible to achieve up to 221,03% of reduction in the energy consumption combined with 34,60% of performance loss. This example confirms our assumption that it is possible to significantly decrease the energy consumption on a grid platform without compromising proportionally the performance.

Keywords: High Performance Computing, scheduling algorithms, energy-aware, simulation.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

AMOK	*Advanced Metacomputing Overlay Kit*
APIs	*Application Programming Interfaces*
BBF	*Backfill Best Fit*
BFF	*Backfill First Fit*
BoT	*Bag-of-Tasks*
CPU	*Central Processing Unit*
CT	*Completion Time*
DAG	*Direct Acyclic Graphs*
DFPLTF	*Dynamic Fastest Processor to Largest Task First*
DLECSA	*Dynamic Low Energy Consumption Scheduling Algorithm*
DLECSA2	*Dynamic Low Energy Consumption Scheduling Algorithm Version 2*
DLECSA3	*Dynamic Low Energy Consumption Scheduling Algorithm Version 3*
DN	*Do Nothing*
DVS	*Dynamic Voltage Scaling*
DVFS	*Dynamic Voltage and Frequency Scaling*
EC	*Energy-congruent*
ECI	*Energy Congruent Index*
EDF	*Earliest Deadline First*
EDFPLTF	*Energy Dynamic Fastest Processor to Largest Task First*
ESuff	*Energy Suferrage*
EWQ	*Energy Workqueue*
EXSuff	*Energy XSuferrage*
FIFO	*First In First Out*
FLOPS	*FLoating-point Operations Per Second*
FPLTF	*Fastest Processor to Largest Task First*
GRAS	*Grid Reality and Simulation*
HAMA	*Heterogeneity Aware Meta-scheduling Algorithm*
HF	*High Frequency*
HPC	*High Performance Computing*
LCR	*Less Consuming Resource*
LECSA	*Low Energy Consumption Scheduling Algorithm*
LECSA2	*Low Energy Consumption Scheduling Algorithm Version 2*
LECSA3	*Low Energy Consumption Scheduling Algorithm Version 3*
LF	*Low Frequency*
LIBTS	*Library Tasks Scheduling*
LIFO	*Last In First Out*
MFW	*Mega Flops per Watt*
MIPS	Milhões de Instruções por Segundo

MSG	*Meta-SimGrid*
RAM	*Random Access Memory*
RNA	Redes Neurais Artificiais
RR	*Round-Robin*
SA	*Simple Aggregation of Jobs*
SHF	*Scaled to the Highest supported Frequency*
SJF	*Shortest Job First*
SLF	*Scaled to the Lowest supported Frequency*
SMPI	*Simulated MPI*
SRT	*Shortest Remaining Time Next*
SSQ	*Scaled to the lowest Supported freQuency*
Suff	*Sufferage*
SWQ	*Scaled to the loWest supported freQuency*
TASA	*Thermal-Aware task Scheduling Algorithm*
TBA	*Time to Become Available*
TI	Tecnologia da Informação
WQ	*Workqueue*
WQR	*Workqueue with Replication*
XBT	*eXtended Bundle of Tools*
XML	*eXtensible Markup Language*
XSuff	*XSufferage*

SUMÁRIO

1 INTRODUÇÃO

A evolução rápida de tecnologias computacionais traz uma série de oportunidades e possibilidades de pesquisa em Computação de Alto Desempenho (do inglês, *High Performance Computing* – HPC). Entretanto, em alguns casos, à medida que os computadores adquirem maior capacidade de processamento eles precisam de mais energia para operar. Isto preocupa pelo impacto negativo que o aumento no consumo de energia tem trazido ao meio ambiente, pelo custo financeiro para mantê-los em operação e pela necessidade de reduzir o consumo de energia.

O aquecimento global, as alterações climáticas, o esgotamento das reservas de energia e o custo com energia, são algumas das principais preocupações que inquietam os pesquisadores atualmente. Deste modo, empresas de Tecnologia da Informação (TI) enfrentam o desafio de oferecer maior poder computacional, atendendo às necessidades de expansão das empresas, sem que ocorram perdas significativas com gastos em energia e refrigeração [SCA07]. As situações citadas conduzem pesquisadores, fabricantes e fornecedores de TI a investirem em iniciativas de Computação Verde, do inglês, *Green Computing*, com o intuito de projetar soluções eficientes em termos de eficiência energética e desempenho.

Eficiência energética, em HPC, tornou-se uma questão importante nos últimos anos. Assim, ambientes HPC tornaram-se um cenário fértil para o desenvolvimento de soluções para o problema da eficiência energética. Um exemplo de ambiente de alto desempenho amplamente utilizado é o de grade computacional, que permite o compartilhamento de recursos de *hardware* e de *software* de forma transparente. Em função dessa transparência, os usuários não devem estar cientes da escalabilidade, heterogeneidade e localização geográfica dos recursos disponíveis na grade. Além disso, as grades computacionais possibilitam a alocação de uma enorme quantidade de recursos para a execução de aplicações paralelas a um custo menor do que se fossem utilizados supercomputadores [SIL03].

Recentemente, alguns trabalhos têm sido desenvolvidos para propor soluções que resolvam o problema da eficiência energética em ambientes HPC (ver Capítulo 3). A maioria das abordagens propostas para reduzir o consumo de energia, baseia-se em: ou desligar recursos ociosos, ou técnicas específicas de escalonamento de tarefas ou ainda alteração da frequência de operação da unidade de processamento. Este trabalho apresenta uma abordagem para melhorar a eficiência energética em grades computacionais, através do uso de algoritmos de escalonamento energeticamente eficientes com gestão inteligente de recursos ociosos. Nosso objetivo principal é reduzir o consumo de energia na execução de tarefas, sem reduzir de forma significativa o desempenho da aplicação. As principais contribuições deste trabalho podem ser resumidas como segue:

- Desenvolvimento de um modelo genérico de consumo de energia para grades computacionais;
- Desenvolvimento de algoritmos de escalonamento energeticamente eficientes, que sejam capazes de trabalhar com a gestão das máquinas ativas e controle de ociosidade;
- Avaliação do comportamento de algoritmos de escalonamento para grades computacionais em relação ao consumo de energia e desempenho.

Para avaliar a abordagem proposta por este trabalho, simulou-se um ambiente de grade usando o *framework* SimGrid (ver Seção 2.3) em conjunto com a biblioteca LIBTS (*Library Tasks Scheduling*) (ver Seção 2.4). Optou-se por simulação, em virtude da dificuldade de utilizar-se um ambiente real de grade, que possui um custo elevado, configuração complexa e dificuldade de repetição de determinados testes. Quanto ao tipo de aplicações submetidas à grade, consideramos o tipo mais utilizado em ambientes de grade, as aplicações *bag-of-tasks* (aplicações nas quais as tarefas são independentes, sem comunicação entre si). Essas aplicações são consideradas adequadas, uma vez que em tais ambientes o custo da comunicação é bastante significativo, devido ao fato dos recursos serem interligados por grandes redes de computadores que possuem uma latência de comunicação alta.

1.1 Motivação e Objetivos

O excesso de consumo de energia é um problema em ambientes de grande porte, como é o caso das grades computacionais. Assim, focar apenas no desenvolvimento de algoritmos de escalonamento direcionados para melhorias de desempenho, não é mais suficiente. Por essa razão, propor estratégias de escalonamento de tarefas para reduzir o consumo de energia em grades computacionais motiva este trabalho. Uma vez que o uso de um bom algoritmo de escalonamento pode evitar, por exemplo: que máquinas fiquem ociosas desnecessariamente e/ou com cargas desbalanceadas e que tarefas sejam enviadas para máquinas inadequadas, diminuindo assim os gastos desnecessários com energia.

O objetivo principal deste trabalho é contribuir para as iniciativas de *Green Computing* no que diz respeito à redução do consumo de energia na execução de aplicações em ambientes de grade computacional. Através do desenvolvimento de algoritmos de escalonamento de tarefas energeticamente eficientes que não ocasionem perdas significativas de desempenho.

A presente Dissertação possui ainda os seguintes objetivos estratégicos:

- Domínio das técnicas de escalonamento de tarefas para grades computacionais;
- Domínio da ferramenta de simulação SimGrid;

- Conhecimento das técnicas atualmente utilizadas na redução de consumo de energia em ambientes de HPC;
- Conhecimento sobre os cenários de testes necessários para validar algoritmos de escalonamento.

Os objetivos específicos por sua vez, incluem:

- Desenvolvimento do módulo de cálculo de consumo de energia na execução de aplicações;
- Desenvolvimento de algoritmos de escalonamento energeticamente eficientes estáticos e dinâmicos;
- Modificações nos algoritmos específicos para grades: *Workqueue* (WQ), *Suferrage*, *XSufferage* e *Dynamic Fastest Processor to Largest Task First* (DFPLTF) para que passem a operar voltados à eficiência energética e não apenas desempenho;
- Avaliação do desempenho e da eficiência energética dos algoritmos de escalonamento citados no item anterior, bem como dos desenvolvidos neste trabalho.

1.2 Estrutura do Documento

A estrutura deste documento está organizada como segue. No Capítulo 2, apresenta-se uma contextualização sobre: o ambiente de grade computacional, o tema escalonamento de tarefas e os algoritmos mais conhecidos na área, a ferramenta de simulação SimGrid e a biblioteca LIBTS. Em seguida, o Capítulo 3 apresenta os trabalhos relacionados aos temas economia de energia e escalonamento de tarefas em ambientes HPC. O Capítulo 4 detalha a implementação do módulo de cálculo de consumo de energia e dos algoritmos de escalonamentos voltados à eficiência energética desenvolvidos neste trabalho. No Capítulo 5, apresentam-se os resultados obtidos e a respectiva discussão dos mesmos. Por fim, o Capítulo 6 encerra esta Dissertação, com as conclusões e direções para trabalhos futuros.

2 REFERENCIAL CONCEITUAL

O objetivo deste capítulo é explicar o ambiente de grade computacional para o qual os algoritmos de escalonamento energeticamente eficientes desenvolvidos por este trabalho são direcionados. Também são apresentados alguns algoritmos de escalonamento tradicionais e alguns específicos para grades computacionais. Como é utilizada simulação para construir um ambiente de grade computacional, pela impossibilidade de trabalhar com um ambiente real por ser demorado e custoso, será explicado o funcionamento do robusto ambiente de simulação SimGrid. Por fim, é apresentada a *Library Tasks Scheduling* (LIBTS), pois apesar de o SimGrid ter sido criado para trabalhar com algoritmos de escalonamento em aplicações científicas paralelas, ele não possui nenhum algoritmo implementado internamente. Neste cenário, foi desenvolvida a biblioteca LIBTS que implementa diversos algoritmos de escalonamento para o SimGrid.

2.1 Grades Computacionais

Grade computacional, de acordo com [REI05], é uma infraestrutura composta por *hardware* e *software* que permite o compartilhamento de recursos, tais como: dados, capacidade de processamento e armazenamento, sendo que esses recursos podem estar espalhados geograficamente. Além disso, as grades podem ser compostas por uma grande variedade de recursos, como: estações de trabalho, supercomputadores, *clusters*, instrumentos científicos, dentre outros. Segundo [SIL03], o objetivo principal das grades é possibilitar a alocação de uma enorme quantidade de recursos para a execução de aplicações paralelas a um custo menor do que se fossem utilizados supercomputadores.

De acordo com [BUY02], as grades computacionais podem ser classificadas em função de sua funcionalidade em:

- Grades de processamento: são utilizadas para executar aplicações que precisam de uma grande capacidade computacional, voltadas a resolver problemas que não poderiam ser resolvidos por um único recurso;
- Grades de dados: são utilizadas para gerenciar o armazenamento e acesso a grandes quantidades de dados;
- Grades de serviços: provê uma grande quantidade de serviços, tais como: *softwares* e recursos computacionais.

Como construir uma grade que possua essas três funcionalidades é muito difícil e custoso, na maioria das vezes opta-se por construir uma grade computacional com a funcionalidade mais apropriada para a necessidade apresentada.

A seguir, são introduzidas as características de uma grade computacional segundo [SIL03]:

- Heterogeneidade dos recursos: essa característica dificulta o escalonamento de tarefas em grades, pois os recursos podem ter velocidade de processadores diferente, interconexão diferente, velocidade de memória diferente, velocidade e tamanho de disco diferente, dentre outros. Diferenças que podem fazer com que algumas aplicações não sejam apropriadas para determinados recursos;
- Compartilhamento de recursos: como em uma grade pode ocorrer variações na carga e na disponibilidade das máquinas, pode ocorrer sobrecarga em alguns recursos, prejudicando o desempenho de algumas aplicações;
- Movimentação de dados: pelo fato das aplicações que executam em grades terem uma grande quantidade de dados que precisam ser movidos de um lugar para o outro, é preciso considerar o tempo gasto com a transferência desses dados.

Em função das características mencionadas anteriormente, escalonamento eficiente de tarefas em um ambiente de grade é um problema complexo. Para obter um bom desempenho na execução das tarefas, o algoritmo utilizado para distribuí-las necessitará escolher os recursos mais apropriados. Outro fator que deve ser considerado quando se pretende trabalhar com grades computacionais, é o tipo de aplicação mais apropriado para executar nesse ambiente. Devido ao fato de que normalmente os recursos de uma grade são interligados por grandes redes de computadores, que possuem uma latência de comunicação alta, as aplicações que forem submetidas à grade, normalmente são do tipo *Bag-of-Tasks* (BoT). Esse tipo de aplicação possui tarefas independentes, que não necessitam trocar informações entre si, ou seja, não há comunicação ou dependências entre tarefas. Aplicações BoT incluem buscas maciças (tais como quebra de chave), aplicações de manipulação de imagem e algoritmos de mineração de dados [SIL09].

Na próxima seção falaremos a respeito de escalonamento de tarefas e sobre alguns dos principais algoritmos de escalonamento específicos para grades computacionais.

2.2 Escalonamento de Tarefas

Segundo [TAN08], o escalonador é responsável por decidir qual *job* irá executar primeiro, caso hajam vários *jobs* prontos para executar competindo pelo uso da *Central Processing Unit* (CPU). Sendo 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 Seção 2.2.1, é apresentada a taxonomia hierárquica para

algoritmos de escalonamento proposta por *Casavant* e *Kuhl* [CAS88] e na Seção 2.2.2 são abordados alguns dos principais algoritmos de escalonamento conhecidos.

2.2.1 Taxonomia Hierárquica para Algoritmos de Escalonamento

Visando criar uma terminologia comum que facilite o entendimento dos diversos cenários nos quais os algoritmos de escalonamento podem ser aplicados, *Casavant* e *Kuhl* criaram uma taxonomia hierárquica para algoritmos de escalonamento em sistemas computacionais distribuídos, ilustrada na Figura 1. Em uma grade computacional, normalmente o tipo de escalonamento necessário encontra-se no ramo Global.

Figura 1 – Taxonomia hierárquica para algoritmos de escalonamento em sistemas computacionais paralelos e distribuídos. Adaptado de [CAS88].

A seguir é apresentada uma explicação sobre a classificação proposta por *Casavant* e *Kuhl* [CAS88]*:*

- **Local *versus* Global:** O escalonamento local determina como os processos residentes em um único processador são alocados e executados. Já o escalonamento global utiliza informações sobre o sistema para alocar processos para múltiplos processadores.
- **Estático *versus* Dinâmico:** No escalonamento estático, as informações são obtidas antes do início do escalonamento da aplicação, sem possuir informações sobre as mudanças dinâmicas de estado do sistema no decorrer do processo. Ou

seja, no escalonamento estático a atribuição de tarefas às máquinas é feita antes do início da execução. Já no escalonamento dinâmico é realizada a alocação de tarefas durante a execução da aplicação.

- **Ótimo *versus* Sub-ótimo:** se toda a informação do estado dos recursos e da aplicação é conhecida, o escalonamento poderá ser ótimo. Nos casos em que é computacionalmente inviável obter tais informações, o escalonamento alcançado é considerado sub-ótimo.
- **Aproximação *versus* Heurística:** o algoritmo de aproximação procura implementar uma solução que seja suficientemente boa em relação a que foi definida como ótima. Já o algoritmo de heurística, leva em consideração alguns parâmetros que afetam o sistema de uma maneira indireta, como por exemplo, a comunicação entre processos.
- **Distribuído *versus* Não distribuído:** nos escalonadores dinâmicos, as decisões de escalonamento global podem ser de um escalonador centralizado ou pode ser compartilhada por múltiplos escalonadores distribuídos. Já a estratégia centralizada pode ser mais simples em termos de implementação, se comparada à distribuída. Entretanto, poderá ser um gargalo em termos de desempenho, pois muitas aplicações podem ser enviadas para serem escalonadas simultaneamente.
- **Cooperativo *versus* Não Cooperativo:** no modo cooperativo, cada escalonador responsabiliza-se por carregar sua própria porção do escalonamento de tarefas, mas todos os escalonadores trabalham em conjunto para o sistema global. Esta situação não ocorre no modo não cooperativo, onde os escalonadores individuais agem de forma independente, não se preocupando em melhorar o desempenho do resto do sistema.

Os algoritmos de escalonamento adicionalmente também podem ser classificados em [CAS88] [NEM11]:

- **Preemptivos e Não Preemptivos:** escalonadores preemptivos permitem que uma tarefa em execução seja interrompida temporariamente e posteriormente retomada. Esse fato pode ocorrer se uma tarefa com maior prioridade chegar para ser executada. Já nos escalonadores não preemptivos, uma vez que uma tarefa for iniciada ela será executada até sua conclusão.
- **Homogêneos e Heterogêneos:** escalonadores que operam em sistemas homogêneos são aqueles nos quais as tarefas são distribuídas pelas unidades de processamento que possuem a mesma capacidade de processamento, memória e disponibilidade de recursos. Já os escalonadores que operam em sistemas heterogêneos, são capazes de lidar com sistemas nos quais as unidades têm diferentes capacidades computacionais.

Essa taxonomia além de auxiliar na organização dos algoritmos de escalonamento também facilita o seu desenvolvimento, pois quanto mais informações forem conhecidas sobre o tipo de ambiente e de aplicação com os quais se trabalhará, maior será a eficiência do algoritmo desenvolvido.

2.2.2 Algoritmos de Escalonamento

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 [TAN08]:

- ***First In First Out* (FIFO):** algoritmo de escalonamento não preemptivo, no qual os processos recebem tempo de CPU na mesma ordem em que 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 processos 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, que assume que os tempos de execução dos *jobs* são conhecidos antecipadamente. Caso haja várias tarefas em uma fila de entrada, de igual importância para serem executadas, o escalonador irá selecionar a tarefa mais curta primeiro.

Figura 2 – Exemplo de escalonamento com a tarefa mais curta primeiro. (a) Executando 4 *jobs* na ordem original. (b) Executando 4 *jobs* na ordem do *job* mais curto primeiro. Extraído de [TAN08].

Se os *jobs* forem executados como mostra a Figura 2(a), o tempo de resposta para a tarefa A será de 8 unidades de tempo, da B 12 unidades de tempo, da C 16 unidades de tempo e da D 20 unidades de tempo, gerando uma média de 14 unidades de tempo. Entretanto, se os *jobs* forem executados como mostra a Figura 2(b), o tempo de resposta para a tarefa B será de 4 unidades de tempo, da C 8 unidades de tempo, da D 12 unidades de tempo e da A 20 unidades de tempo, gerando uma média de 11 unidades de tempo, o que demonstra o ganho de desempenho ao utilizar o algoritmo SJF em relação ao FIFO.

- ***Shortest Remaining Time Next* (SRT):** no algoritmo SRT o escalonador escolhe o *job* cujo tempo de execução restante é o mais curto. Ou seja, ao chegar um novo *job* o seu tempo será comparado com o tempo que resta do j*ob* atual. Caso o novo

job precise de menos tempo para terminar do que o *job* atual, este será suspenso e o novo *job* será executado.

- ***Round-Robin* (RR):** o algoritmo RR opera em sistemas iterativos, realizando um rodízio entre os processos da seguinte maneira: cada *job* possui um intervalo de tempo (*quantum*) durante o qual ele pode ser executado. Caso ele esteja em execução e seu *quantum* terminar (esse *job* será colocado no final da fila), será feita a preempção da CPU e esta será alocada para outro *job*.
- ***Earliest Deadline F*irst (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 BBF 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.

Como citado na seção anterior, o escalonamento em grades computacionais é um processo complexo, em virtude disso existem alguns algoritmos específicos para esse tipo de ambiente, tais como:

- ***Workqueue* (WQ)**: neste algoritmo, a escolha de qual tarefa será submetida para execução é feita de maneira aleatória sempre que um recurso ficar disponível. O objetivo é que um maior número de tarefas seja atribuído para máquinas mais rápidas, fazendo com que as máquinas mais lentas executem cargas leves. Uma vantagem deste algoritmo é que ele não precisa de informações a respeito das tarefas ou dos recursos. Entretanto, se uma tarefa grande for atribuída a um processador lento próximo do final da execução da aplicação, a sua conclusão será adiada até que essa tarefa seja finalizada [SIL03].
- ***Workqueue with Replication* (WQR):** inicialmente o WQR é similar ao WQ, ou seja, as tarefas são enviadas para executarem nas máquinas que estão disponíveis. No momento que uma máquina finaliza a execução de uma tarefa, ela recebe uma nova tarefa. A diferença entre o WQR e o WQ ocorre quando uma máquina se torna disponível e não há mais nenhuma tarefa na fila para ser executada. Neste momento o WQR inicia a replicação das tarefas que ainda estão em execução. Assim que a tarefa original ou uma de suas réplicas finalizarem a execução, as outras são interrompidas [SIL03].

- ***Sufferage* (Suff)** [CAS00] [FRA11]: a lógica deste algoritmo consiste em determinar o quanto cada tarefa pode ser prejudicada se não for escalonada no processador que a execute de forma mais eficiente. O valor *sufferage* de cada tarefa é a diferença entre o melhor e o segundo melhor *Completion Time* (CT), considerando todos os processadores da grade. A tarefa com o maior valor s*ufferage* terá prioridade de execução. O valor de CT é dado pela fórmula *CT = Time to Become Available (TBA) + Task Cost.* Onde, TBA é o tempo para o *host* tornar-se disponível, *Task Cost = (Task size / Host speed) / (1 - Host load)*, onde:
    - *Host Speed*: representa a velocidade da máquina.
    - *Host Load*: representa a fração de CPU da máquina que não está disponível para a aplicação, ou seja, a fração de CPU que está sendo usada por outras aplicações. O *Host Load* varia com o tempo, dependendo da carga de trabalho da máquina.
    - *Task Size*: é o tempo necessário para uma máquina com *Host Speed = 1* completar a tarefa quando *Host Load = 0*.

  Como o valor *sufferage* de cada tarefa mudará durante a execução da aplicação, cada vez que uma tarefa terminar, as demais, que ainda não começaram a execução, serão desalocadas e o algoritmo calculará os valores *sufferage* atuais. Esse processo é repetido até que todas as tarefas sejam completadas. Uma desvantagem do *Sufferage* é que ele precisa conhecer informações sobre as tarefas e os recursos e tais informações nem sempre estão disponíveis.
- ***XSufferage* (XSuff)** [CAS00] [FRA11]: o *XSufferage* é uma modificação do *Sufferage*, cuja principal diferença é o método usado para calcular o valor do *sufferage*. O *XSufferage* considera a transferência dos dados de entrada da tarefa durante o cálculo dos tempos de execução. Dessa maneira, 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 quando vai escalonar uma tarefa.
- ***Dynamic Fastest Processor to Largest Task First* (DFPLTF)**: é a versão dinâmica do algoritmo estático FPLTF [MEN95]. O DFPLTF também necessita de três informações para escalonar tarefas: *Task Size*, *Host Load* e *Host Speed*. Quando o algoritmo inicia, o TBA de cada *host* é iniciado com 0 e as tarefas são ordenadas por tamanho em ordem decrescente. Desta forma, a maior tarefa é a primeira a ser alocada para o *host* que provê o menor CT. Assim que uma tarefa é alocada para uma máquina, o valor do TBA relativo a este *host* é incrementado

com *Task Cost*. As tarefas são alocadas até que todas as máquinas da grade fiquem em uso. Depois disso, a execução da aplicação é iniciada. No momento que uma tarefa é completada, todas as demais que não estão executando são escalonadas novamente até que todas as máquinas fiquem em uso. Tal processo é repetido até que todas as tarefas sejam completadas.

Segundo [REI05], 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, 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.

2.3 SimGrid

No presente trabalho, utiliza-se a ferramenta de simulação de aplicações em ambientes distribuídos heterogêneos, SimGrid [SIM12], 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: código *opensource*, possui uma boa documentação (com descrições sobre os módulos da ferramenta e seu uso) e está disponível para plataforma Linux, Windows e MacOS. Além disso, diversos trabalhos na área de escalonamento de tarefas, em ambientes distribuídos, utilizam o SimGrid como ferramenta de simulação.

Antes de detalhar a estrutura e o funcionamento do SimGrid é preciso explicar o motivo de utilizar simulação ao invés de um ambiente real. No trabalho de [MUR02] são citados alguns fatores para a utilização de simulação:

- A configuração de um ambiente de teste real demanda muito tempo, é custosa e precisa de recursos intensivos;
- Um ambiente real não fornece um ambiente repetível e controlável para a experimentação e avaliação de estratégias de escalonamento;
- Aplicações reais demoram longos períodos de tempo para executar, dessa forma, não seria viável executar um grande número de experimentos;

- A utilização de recursos reais faz com que seja difícil explorar uma grande variedade na configuração de recursos;
- Ao analisar novos modelos e algoritmos é preciso utilizar um grande número de testes, o que envolve muitos recursos e torna esse processo de alto custo econômico, se for realizado em uma plataforma real.

Em decorrência dos fatores mencionados anteriormente, a simulação pode ser usada de maneira eficiente para demonstrar como um sistema real se comporta.

2.3.1 Arquitetura

A Figura 3 mostra a arquitetura do SimGrid, que é composta pelas seguintes camadas: ambientes de programação, núcleo de simulação e base [SIM12].

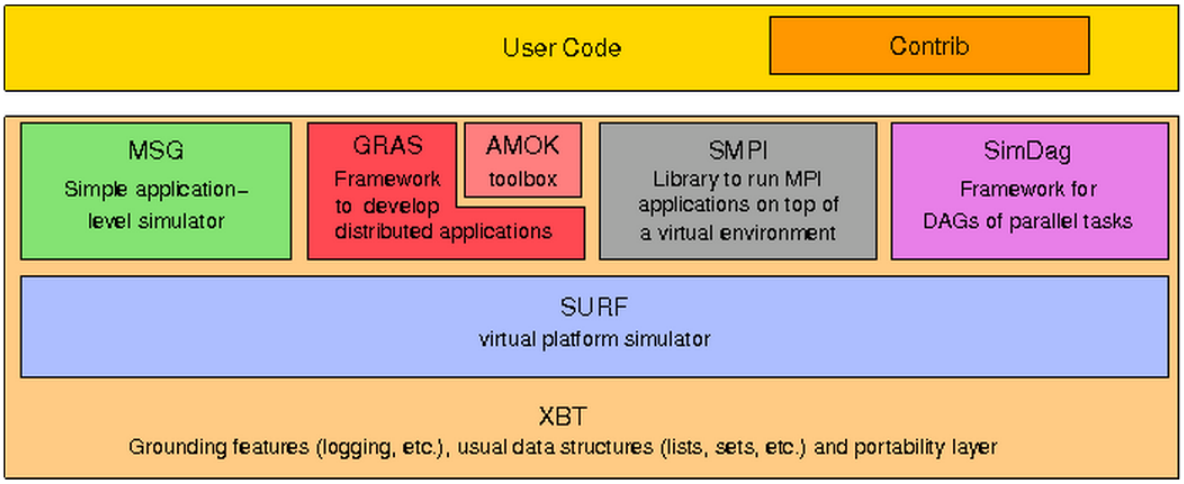

Figura 3 – Componentes do SimGrid. Extraído de [SIM12].

Ambientes de Programação: o SimGrid possui diversos ambientes de programação, cada qual com um objetivo específico. Estes ambientes serão descritos a seguir:

- *Meta-SimGrid* (MSG): deve ser utilizado para modelar problemas teóricos e para comparar diferentes heurísticas;
- *Simulated MPI* (SMPI): utilizado para simular o comportamento de aplicações MPI;
- *Grid Reality and Simulation* (GRAS): viabiliza a execução de aplicações reais para estudos e testes;
- *Advanced Metacomputing Overlay Kit* (AMOK): implementa em alto nível diversos serviços necessários para várias aplicações distribuídas;
- SimDag: usado para a simulação de aplicações paralelas, utilizando um *Direct Acyclic Graphs* (DAG). Nesse ambiente é possível especificar relações de dependência entre tarefas de um programa paralelo.

Núcleo de simulação: composta pelo módulo SURF, que simula uma plataforma virtual. Como é de muito baixo nível, não se destina a ser usado pelos usuários comuns.

Base: composta pelo *eXtended Bundle of Tools* (XBT), que é uma biblioteca portável que oferece alguns serviços como suporte de registro, de exceção e de configuração. O XBT também é responsável por dar suporte à portabilidade da ferramenta.

2.3.2 Implementação e Documentação

O SimGrid é implementado em linguagem C, *opensource* e funciona em modo texto. Está disponível para ambientes Linux, Windows e MacOS. A simulação realizada pelo SimGrid é determinística, ou seja, repetindo a mesma simulação os resultados retornados serão sempre os mesmos.

A documentação do SimGrid pode ser consultada no site oficial do projeto [SIM12], sendo bem estruturada e de fácil compreensão.

2.3.3 Modelagem da Plataforma e das Tarefas da Grade

Para realizar simulações no SimGrid é preciso modelar três arquivos. O primeiro contendo a plataforma da grade, o segundo contendo tarefas e o terceiro é um arquivo executável compilado em C usando as *Application Programming Interfaces* (APIs) do SimGrid. O arquivo que configura a plataforma da grade (*plataform.xml*) especifica todo o conjunto de recursos da grade, assim como a estrutura de interconexão entre eles. O mesmo, pode ser modelado no formato *eXtensible Markup Language* (XML), contendo as seguintes informações:

- 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.

O arquivo de modelagem de tarefas que usa o ambiente de programação MSG (ambiente utilizado neste trabalho) especifica os processos que serão executados em cada recurso. Tal arquivo possui a identificação dos *hosts*, com os mesmos nomes do arquivo *plataform.xml* e suas funções de *maste*r ou *slave*. O *host* que tem a função de *master* possui as seguintes informações: número de tarefas que receberá, tamanho de computação das tarefas, tamanho de comunicação das tarefas e a identificação de quem são os *slaves*. O *host* que possui a função de *slave* não possui nenhuma informação.

A Figura 4 mostra o arquivo de configuração das tarefas. Já a Figura 5 apresenta parte do arquivo de configuração da plataforma.

```
<platform version="3">
  <!-- The master process (with some arguments) -->
  <process host="Tremblay" function="master">
     <argument value="20"/>       <!-- Number of tasks -->
     <argument value="50000000"/>  <!-- Computation size of tasks -->
     <argument value="1000000"/>   <!-- Communication size of tasks -->
     <argument value="Jupiter"/>  <!-- First slave -->
     <argument value="Fafard"/>   <!-- Second slave -->
     <argument value="Ginette"/>  <!-- Third slave -->
     <argument value="Bourassa"/> <!-- Last slave -->
     <argument value="Tremblay"/> <!-- Me! I can work too! -->
  </process>
  <!-- The slave process (with no argument) -->
  <process host="Tremblay" function="slave"/>
  <process host="Jupiter" function="slave"/>
  <process host="Fafard" function="slave"/>
  <process host="Ginette" function="slave"/>
  <process host="Bourassa" function="slave"/>
</platform>
```

Figura 4 – Arquivo de configuração da aplicação a ser simulada.

```
<platform version="3">
<platform version="3">
 <AS  id="AS0"  routing="Full">
  <!-- ljlkj -->
  <host id="Tremblay" power="105400000000"/>
  <host id="Jupiter" power="9630000000"/>
  <host id="Fafard" power="17270000000"/>
  <host id="Ginette" power="10320000000"/>
  <host id="Bourassa" power="7030000000"/>
  <link id="6" bandwidth="41279125" latency="5.9904e-05"/>
  <link id="11" bandwidth="252750" latency="0.00570455"/>
  <link id="3" bandwidth="34285625" latency="0.000514433"/>
  <link id="7" bandwidth="11618875" latency="0.00018998"/>
  <link id="9" bandwidth="7209750" latency="0.001461517"/>
  <link id="12" bandwidth="1792625" latency="0.007877863"/>
  <link id="2" bandwidth="118682500" latency="0.000136931"/>
  <link id="8" bandwidth="8158000" latency="0.000270544"/>
  <link id="1" bandwidth="34285625" latency="0.000514433"/>
  <link id="4" bandwidth="10099625" latency="0.00047978"/>
  <link id="0" bandwidth="41279125" latency="5.9904e-05"/>
  <link id="10" bandwidth="4679750" latency="0.000848712"/>
  <link id="5" bandwidth="27946250" latency="0.000278066"/>
  <link id="loopback" bandwidth="498000000" latency="0.000015" sharing_policy="FATPIPE"/> <!—FATPIPE
 ou SHARED -->
  <route src="Tremblay" dst="Tremblay"><link_ctn id="loopback"/></route>
  <route src="Jupiter" dst="Jupiter"><link_ctn id="loopback"/></route>
  <route src="Fafard" dst="Fafard"><link_ctn id="loopback"/></route>
  <route src="Ginette" dst="Ginette"><link_ctn id="loopback"/></route>
  <route src="Bourassa" dst="Bourassa"><link_ctn id="loopback"/></route>
  <route src="Tremblay" dst="Jupiter">
    <link_ctn id="9"/>
  </route>
  <route src="Tremblay" dst="Fafard">
    <link_ctn id="4"/><link_ctn id="3"/><link_ctn id="2"/><link_ctn id="0"/><link_ctn id="1"/>
<link_ctn id="8"/>
  </route>
  <route src="Tremblay" dst="Ginette">
    <link_ctn id="4"/><link_ctn id="3"/><link_ctn id="5"/>
  </route>
  <route src="Tremblay" dst="Bourassa">
    <link_ctn id="4"/><link_ctn id="3"/><link_ctn id="2"/><link_ctn id="0"/><link_ctn id="1"/>
<link_ctn id="6"/><link_ctn id="7"/>
  </route>
…
</AS>
</platform>
```

Figura 5 – Arquivo de configuração da plataforma de simulação.

Para que a simulação ocorra, é necessário que algumas funções próprias do SimGrid sejam implementadas, tais como: *master*, *slave* e escalonamento (algoritmo de escalonamento que atribui às tarefas aos nós). Essas funções estão implementadas no arquivo mostrado pela Figura 6.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "escalonamento.h" /* LIBTS */

XBT_LOG_NEW_DEFAULT_CATEGORY(msg_test,"Messages specific for this msg example");
  int master(int argc, char *argv[]);
  int slave(int argc, char *argv[]);
void escalonamento(todo, number_of_tasks, slaves, slaves_count);
MSG_error_t test_all(const char *platform_file, const char *application_file);
```

Figura 6 – Arquivo de simulação do SimGrid.

Segundo [SIL09], cada tarefa passa por três fases durante a sua execução: inicialização, computação e conclusão. No SimGrid essas três fases podem ser descritas da seguinte forma:

- Fase de inicialização: a tarefa é enviada do mestre para o nó escravo e a tarefa é iniciada. Essa fase inclui a sobrecarga gerada pelo mestre para iniciar uma transferência de dados para um escravo;
- Fase de computação: o escravo processa a tarefa. Qualquer sobrecarga relacionada com a recepção de arquivos de entrada por um nó escravo também está incluída nesta fase;
- Fase de conclusão: quando a tarefa estiver concluída, o mestre recebe uma mensagem de que o escravo pode receber outra tarefa. A fase de inicialização de um escravo pode ocorrer concomitantemente com a fase de conclusão de outro nó escravo.

2.4 *Library Tasks Scheduling* (LIBTS)

Segundo [FRA11], apesar de o 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. A LIBTS é desenvolvida em linguagem C e implementa os seguintes algoritmos de escalonamento: FIFO, LIFO, RR, SJF, WQ, WQR, *Sufferage*, *XSufferage* e DFPLTF.

Para interagir com os módulos do SimGrid, a LIBTS foi adicionada ao módulo MSG. Sua estrutura é composta pelo módulo principal chamado *escalonamento* que possui a chamada para todos os algoritmos de escalonamento citados anteriormente. Tais

algoritmos seguem o padrão citado na Seção 2.2.2 com algumas adaptações para poder utilizar a estrutura de dados do MSG.

A LIBTS funciona com aplicações do tipo *Master-Slave*, sendo que o código *masterslave.c* utilizado pela LIBTS foi criado tendo como base o *masterslave_bypass.c* disponibilizado pelo SimGrid. Dentro deste arquivo, são realizadas algumas ações como: a declaração das bibliotecas do SimGrid, dos dados das aplicações e da plataforma, a criação das tarefas, dentre outras. No *masterslave.c* da LIBTS o escalonamento é feito através da chamada da função *escalonamento*, na qual o usuário pode escolher o algoritmo de escalonamento que deseja. Já no *masterslave_bypass.c* o algoritmo é implementado diretamente.

A arquitetura do SimGrid com a LIBTS pode ser vista na Figura 7 [FRA11].

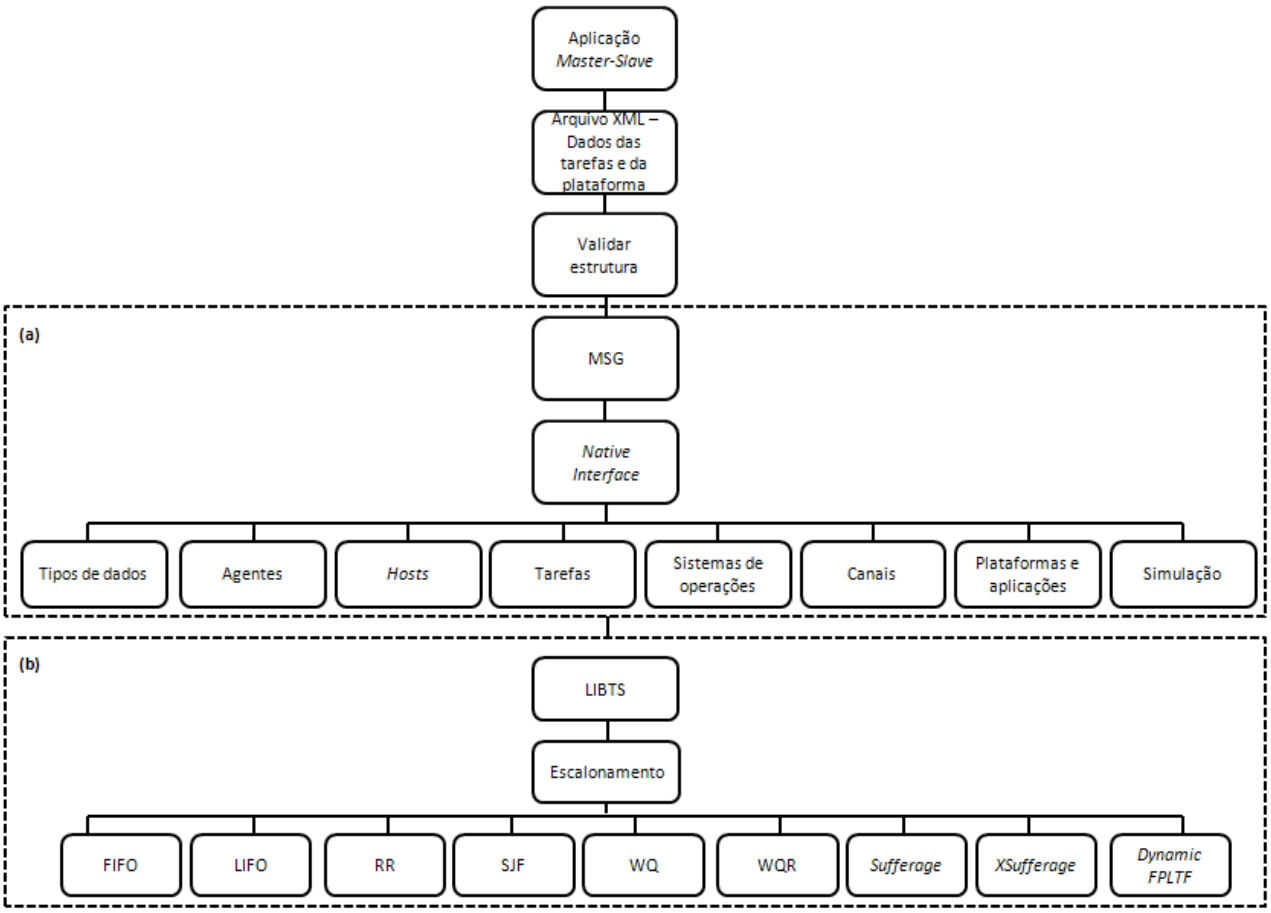

Figura 7 – Arquitetura de uma aplicação no SimGrid. (a) Arquitetura do módulo MSG. (b) Arquitetura da LIBTS. Adaptado de [FRA11].

Caso os usuários queiram desenvolver seu próprio algoritmo de escalonamento para executar na LIBTS, eles podem fazê-lo seguindo os padrões do MSG e em seguida adicionar a chamada da sua função de escalonamento ao código fonte *escalonamento.c*, da seguinte forma:

escalona_nomeAlgoritmo (task, number_of_tasks, slaves, slaves_count)

Depois de ter-se apresentado o referencial conceitual no qual este trabalho fundamenta-se, o próximo capítulo apresenta os trabalhos relacionados à eficiência energética em ambientes de alto desempenho. Eficiência energética é um tema que vem destacando-se na área computacional, devido à dificuldade de manter determinadas infraestruturas computacionais em operação, pelo alto custo associado a gastos com estruturas de refrigeração, consumo de energia e manutenção.

3 TRABALHOS RELACIONADOS

De acordo com [MAM11], para garantir a eficiência energética em um ambiente de grade computacional é preciso dar atenção aos seguintes fatores:

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

A técnica de DVFS é usada para controlar a potência da CPU, pois segundo [MAM11], o consumo de energia do processador é 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. Mas, devido ao fato desta solução ser dependente da carga de trabalho, o desafio, de acordo com [MAM11], 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 [LIU10], 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.

Neste capítulo é apresentado o estado-da-arte em técnicas para a redução do consumo de energia em sistemas computacionais como grades e *clusters*. 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 computacionais e *clusters* apresentam soluções que utilizam controle de recursos ociosos e escalonamento de tarefas para obter eficiência energética.

3.1 Controle de Recursos Ociosos

Essa estratégia consiste em manipular os recursos computacionais disponíveis para que eles não fiquem ociosos consumindo energia de maneira desnecessária. A seguir, são apresentados alguns trabalhos que utilizam tal estratégia.

3.1.1 Ponciano et al.

O trabalho de [PON10] apresenta duas estratégias para economizar energia em grades computacionais oportunistas (nesse tipo de sistema se os recursos computacionais não estiverem sendo utilizados pelo seu “dono”, podem ser usados para executar tarefas de terceiros que tenham sido submetidas à grade), são elas:

- Sobreaviso, do inglês *standby*: mantém ativa a memória *Random Access Memory* (RAM) e reduz a atividade do disco rígido e do processador;
- Hibernação, do inglês *hibernate*: salva o estado da memória RAM no disco rígido e reduz o uso de energia da RAM, do disco rígido e do processador.

Segundo os autores, o tempo gasto para colocar e retirar um computador do estado sobreaviso é menor do que do estado hibernação. Entretanto, o estado de sobreaviso apresenta maior consumo de energia, pelo fato da memória RAM permanecer energizada [PON10].

Um fator que deve ser considerado é que o uso das estratégias de sobreaviso e de hibernação, pode reduzir a vida útil dos recursos da grade caso elas ocasionem um aumento excessivo no número de transições entre o estado ativo e os estados citados acima. Isso ocorre devido ao fato de alguns componentes do computador, como o disco rígido, por exemplo, tolerarem um número máximo de transições durante o seu tempo de vida [PON10]. Outro fator importante é determinar após quanto tempo de inatividade do recurso da grade a estratégia deve ser acionada. Segundo os autores, ao utilizar-se um tempo de inatividade pequeno, o recurso será adormecido logo depois de se tornar ocioso, o que pode aumentar a economia de energia, entretanto, pode aumentar também o tempo de resposta das aplicações que terão que esperar até que o recurso volte à atividade. Já ao utilizar-se um tempo de inatividade grande, o recurso permanecerá no estado de ociosidade durante mais tempo o que pode reduzir a economia de energia. Um ponto positivo é que pode reduzir também o tempo de resposta das aplicações, as quais não precisarão esperar até que o recurso torne-se ativo novamente.

Os resultados dos testes mostraram que as estratégias de sobreaviso e de hibernação reduzem o consumo de energia em mais de 80%. Essa redução no consumo não causa um grande impacto no tempo de resposta das aplicações, aumentando no máximo em 5%, com o uso de hibernação, e em pouco mais de 1%, com o uso de sobreaviso. Os autores também relatam que quanto maior o valor do tempo de inatividade, menor é a economia de energia, (para os testes foram utilizados cinco valores para o tempo de inatividade: 0, 300, 600, 900 e 1200 segundos). Esse fato ocorre devido ao aumento do tempo em que cada máquina permanece no estado ocioso, esperando a chegada de uma nova tarefa [PON10]. Quanto ao número de transições entre o estado ativo e os estados de baixo consumo de energia, a estratégia de hibernação utilizando 50 máquinas e tempo de inatividade igual a 0, apresentou o maior número, em média 15

transições por máquina, o que equivale a 8 transições por dia. Como o número de transições de um disco rígido SATA, por exemplo, pode ser de até 273 por dia, o uso dessa estratégia não reduziria a vida útil desse componente.

3.1.2 Mämmelä et al.

Em [MAM11], é apresentado um escalonador com consciência energética para *clusters*. O escalonador proposto implementa três algoritmos de escalonamento, E-FIFO, E-BFF e E-BBF, que são versões com reconhecimento de consumo de energia dos tradicionais algoritmos: FIFO, BFF e BBF (Seção 2.2.2).

Nesses algoritmos, com adicional de consciência energética, é utilizado o método de desligar nós ociosos caso haja mais de *T* segundos antes do início da primeira tarefa da fila. O trabalho também apresenta modelos para calcular o consumo de energia de diversos componentes de um computador quando ele está ocioso, tais como: processador, memória, disco rígido, dentre outros. Para tal, os computadores que compõem o *cluster* são considerados homogêneos. Para os testes, foram utilizadas tarefas com diferentes características para que se pudessem obter vários níveis de estresse no sistema para cada tipo de tarefa (CPU, memória), recursos a serem utilizados (número de nós, quantidade de memória necessária), tempo para conclusão de tarefas e quantidade de tarefas criadas.

Baseados nos resultados dos testes, os autores mencionam que o escalonador é capaz de reduzir o consumo de energia entre 6 e 16%, dependendo da carga de trabalho, sem que haja uma perda significativa no tempo de resposta ou aumento do tempo de espera das tarefas. Sendo que o E-BFF e o E-BBF, foram mais eficientes nos testes realizados 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.

3.1.3 Montes et al.

O trabalho de [MON12] descreve um escalonador inteligente que visa economizar energia em um ambiente de grade. Os recursos utilizados são provenientes do Grid’5000. Os autores utilizaram como pressuposto que apesar das máquinas serem diferentes, elas possuíam desempenho e consumo idênticos. Por essa razão, segundo eles, não haveria diferença entre executar um *job* em um conjunto de recursos ou em outro. A seguir, estão descritos os cinco estados em que os recursos da grade podem estar:

- *On*: quando o recurso está executando um *job*. A energia consumida foi considerada de 108 *watts*;
- *Off*: significa que o recurso está desligado e portanto, não está ocupado com *jobs*. A energia consumida foi considerada de 5 *watts*;

- *Idle*: significa que o recurso está ligado esperando pelo recebimento de *jobs* para executar. A energia consumida foi considerada de 50 *watts*;
- *Booting*: significa que o recurso está passando do estado de *Off* para *On*. A energia consumida foi considerada de 110 *watts*;
- *Shutting*: quando um recurso é desligado a partir de *On* ou *Idle*. A energia consumida foi considerada de 110 *watts*.

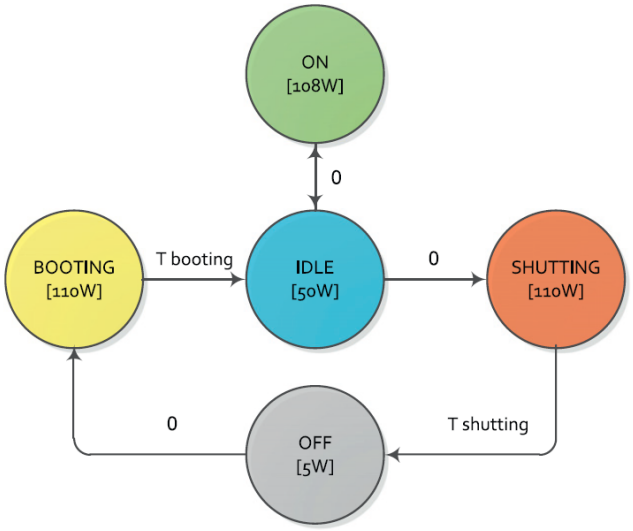

Figura 8 – Ciclo de vida de um recurso. Extraído de [MON12].

A Figura 8 mostra o ciclo de vida dos recursos. O tempo para a passagem de um estado para o outro é apresentado sobre as setas. Os tempos *Tbooting* e *Tshutting* foram estabelecidos em 100 segundos e 10 segundos respectivamente.

Segundo os autores, a atual política do Grid'5000 consistia em deixar os recursos ociosos a espera de novos trabalhos para executar, a fim de satisfazer as necessidades dos usuários rapidamente. Entretanto, essa política chamada de *Always On* era muito ruim em termos de consumo de energia. Em virtude disso, o objetivo dos autores era substituir essa política por novas políticas energéticas, que decidissem entre deixar recursos ligados ou desligá-los assim que eles finalizassem a execução de um *job*. Essas novas políticas foram denominadas [MON12]: *Always On*, *Always Off*, *Switch Off Randomly*, *Load, Switch off* T_s*, Exponential* e *Gamma*. Tais políticas foram organizadas em:

- Não fazer nada, do inglês, *Do Nothing* (DN): consiste em não alterar o recurso de execução do *job* e nem deslocá-lo no tempo para aproveitar recursos que já estão ligados, ou seja, eles são executados conforme definido pelo escalonador;
- Simples agregação de *jobs*, do inglês, *Simple Aggregation of Jobs (SA):* tenta encontrar recursos disponíveis (*Idle*) para executar os novos *jobs*. Dessa maneira, se um *job* for atribuído a um conjunto de recursos que estão *Off* e outros recursos estiverem disponíveis, tanto tempo quanto energia, podem ser salvos.

O cálculo da energia consumida para executar os *jobs* dos experimentos foi baseado nas informações de consumo de energia dos estados: *On*, *Off*, *Idle*, *Booting* e *Shutting*. Analisando os resultados obtidos os autores concluíram que a SA é sempre melhor em termos de economia de energia, chegando a alcançar uma economia de energia, para o *site* da grade localizado em Bordeaux, de 129,254 kWh.

3.2 Escalonamento Voltado para Eficiência Energética

Os trabalhos que serão apresentados nessa seção utilizam a estratégia de escalonamento. Tal estratégia consiste em alocar tarefas a recursos visando reduzir o consumo de energia.

3.2.1 Wang et al.

Um estudo sobre escalonamento de tarefas baseado em previsão de temperatura em um centro de dados é apresentado em [WAN11]. Segundo os autores, altas temperaturas dentro de um centro de dados podem aumentar os custos com resfriamento e as taxas de falha de *hardware*. Segundo [FEN03], a computação em alta temperatura é mais propensa a erros, sendo que a taxa de falha de *hardware* de um dispositivo eletrônico dobra a cada aumento de 10 ºC na temperatura, de acordo com a equação de *Arrenhius*. Como a gestão da carga de trabalho pode reduzir as temperaturas dentro de um centro de dados, o trabalho de [WAN11], baseado em técnicas de previsão de temperatura utilizando Redes Neurais Artificiais (RNA), apresentou um algoritmo de escalonamento de tarefas com reconhecimento térmico, do inglês, *Thermal-Aware task Scheduling Algorithm* (TASA). Tal algoritmo visa reduzir o consumo de energia e a temperatura em um centro de dados, colocando cargas nos nós de computação de forma adequada, reduzindo os picos de temperatura do nó e o tempo de resposta da tarefa.

Para avaliar o desempenho do algoritmo proposto, os autores realizaram um estudo baseado em simulação. O algoritmo de escalonamento utilizado para comparação com o TASA foi o FIFO. De acordo com os autores, quando alguns recursos do *cluster* atingem a temperatura máxima, o TASA atrasa o envio de tarefas, para deixá-los esfriar. Fato que não ocorre com o FIFO, que por não considerar a temperatura, continua a escalonar tarefas para os recursos. Os resultados mostraram que o TASA pode economizar até 13,34% de alimentação do sistema de refrigeração do *cluster*, além de reduzir 2.130Kg de CO_2 por hora, dos cerca de 33.000 kg emitidos a cada hora pelo *cluster* do centro de pesquisa [WAN11]. Outro benefício é que a temperatura máxima do *cluster* diminuiu 6,67 °F. Entretanto, houve um aumento de 15,2% no tempo de resposta das tarefas. Perda que segundo os autores, seria compensada pelos ganhos acima mencionados.

3.2.2 Kim et al.

Em sistemas de *clusters* é de suma importância investir em estratégias de consciência energética, pois o consumo de energia impacta no custo operacional e na confiabilidade desses sistemas [KIM07]. Com o objetivo de minimizar o consumo de energia e cumprir os prazos especificados pelos usuários das aplicações, o trabalho de [KIM07] apresenta algoritmos de escalonamento com consciência energética para aplicações do tipo BoT com restrições de prazo para *clusters* com *Dynamic Voltage Scaling* (DVS) habilitado. DVS é utilizado para reduzir o consumo de energia dinâmica, ajustando a tensão de alimentação de um modo adequado.

O escalonamento proposto funciona da seguinte maneira: quando uma tarefa é recebida, o controlador de recursos decide se aceita o trabalho. O regime de admissão proposto garante que os prazos das tarefas anteriormente aceitas no sistema sejam atendidos. Caso todas as tarefas possam cumprir seus prazos, os elementos de processamento necessários são alocados para a nova tarefa. Os autores descrevem os seguintes passos para a admissão de uma nova tarefa e a sua execução no sistema [KIM07]:

1. Envio de tarefas: um usuário envia uma tarefa do tipo BoT com novo prazo para o sistema de *cluster*;
2. Teste de escalonabilidade e estimativa de energia: o controlador de recursos requer a escalonabilidade e a energia necessária para executar cada tarefa em cada um dos elementos de processamento;
3. Confirmação de escalonabilidade e quantidade de energia: cada elemento de processamento testa a escalonabilidade da nova tarefa e retorna o consumo de energia estimado no caso de ser escalonável;
4. Seleção de elementos de processamento: o controlador de recursos seleciona o elemento de processamento de menor consumo de energia que pode executar cada tarefa.

Como uma tarefa consiste em múltiplas tarefas, os passos de (2) a (4) são repetidos até que todas as tarefas sejam atribuídas aos elementos de processamento. Se todas as tarefas cumprirem os prazos, o controlador de recursos aceita a nova tarefa, senão ele a rejeita, porque não pode garantir o prazo da tarefa.

As simulações foram realizadas com dois algoritmos de escalonamento que são variações do EDF, um que usa DVS, o EDF-DVS e outro com participação proporcional, o EDF-PShare. Para comparar o desempenho, os autores simularam um algoritmo de escalonamento sob tensão mais baixa de alimentação (= 0,9V), e o outro sob tensão mais elevada (= 1,5V).

Os resultados mostraram que os algoritmos com DVS habilitado possuem uma relação de grande aceitação de tarefas, além de consumirem menos energia em comparação com as versões 1.5V-estática:

- O EDF-1.5V sempre executa processadores na velocidade máxima de relógio, por isso possui uma elevada aceitação de tarefas, com o maior consumo de energia, ao ser comparado com o EDF-1.5V, PShare-1.5V, EDF-DVS, PShare-DVS, EDF-0.9V e PShare-0.9V;
- O EDF-1.5V e o PShare-1.5V consomem grande quantidade de energia, porque usam uma tensão de alimentação com 1.5V;
- O EDF-0.9V e o PShare-0.9V mostraram um menor consumo de energia. Mas, eles possuem uma aceitação de tarefas inferior a 40%, mesmo sob condições de baixa sobrecarga.

Os autores também realizaram uma comparação de desempenho entre os algoritmos EDF-DVS *versus* EDF-1.5V e PShare-DVS *versus* PShare-1.5V, em termos de índice de aceitação e consumo de energia. Os resultados mostraram que a melhora na redução de energia sempre é maior do que a degradação da taxa de aceitação. Quando a carga do sistema torna-se baixa, há uma melhoria na economia de energia e uma perda de aceitação baixa. Em resumo, os resultados obtidos mostraram que os algoritmos que usam DVS reduzem o consumo de energia com pouca degradação do prazo das tarefas.

3.2.3 Garg et al.

O trabalho desenvolvido por [GAR09] trata da exploração da heterogeneidade em grades computacionais para a alocação de recursos com uso eficiente de energia. Para reduzir o consumo de energia do sistema, os autores utilizaram a técnica de distribuir as aplicações para serem executadas em paralelo em uma rede em grade. Essa distribuição é feita através da utilização de um algoritmo de escalonamento denominado *Heterogeneity Aware Meta-scheduling Algorithm* (HAMA). O algoritmo HAMA consegue selecionar de maneira energeticamente eficiente os recursos da grade, selecionando primeiro o recurso mais eficiente em termos de energia dentre os disponíveis na grade (os recursos da grade são classificados em ordem de eficiência de energia). HAMA ordena as tarefas recebidas com base no algoritmo EDF (Seção 2.2.2). Em seguida, usa DVS para alterar a velocidade de processamento da CPU com o intuito de alcançar uma maior redução no consumo de energia, sempre buscando respeitar o prazo de conclusão de determinada tarefa.

Os testes foram realizados com base nos seguintes cenários: um contendo variações na urgência para a execução dos *jobs* e o outro com variações na taxa de chegada dos *jobs*. Os resultados dos testes mostraram que o HAMA com uso de DVS pode reduzir o consumo de energia do sistema em até 23%, no pior caso, e em até 50%

no melhor caso, ao ser comparado com o algoritmo EDF. Já ao operar sem DVS, o HAMA obteve uma economia de energia de até 21% [GAR09].

3.2.4 Lammie et al.

Segundo [LAM09], uma carga de trabalho normalmente consiste de muitos processos sequenciais independentes, que podem ter sua execução moldada para satisfazer as restrições de energia. O trabalho fala a respeito de um escalonador que mantém um compromisso entre redução do consumo de energia e pouca perda de desempenho em grades computacionais. Para isso, é utiliza uma variação do número e da frequência dos processadores disponíveis.

No trabalho desenvolvido por [LAM09], são propostas três estratégias que visam minimizar o consumo de energia e maximizar o desempenho, são elas:

- Escala de frequência de CPU: técnica para reduzir a velocidade da CPU, a fim de diminuir a energia consumida e o calor que é dissipado. É empregada para aumentar o rendimento do sistema em vez de ativar nós que serão subutilizados;
- Dimensionamento automático de nós: mecanismo utilizado para ligar e desligar máquinas, a fim de corresponder a requisitos como tamanho da fila e características das tarefas. Máquinas só são ativadas quando se considera que o *cluster* não pode suportar a carga atual e desligadas ao permanecerem inativas por um período de tempo específico;
- Atribuição inteligente de *jobs*: camada de otimização implementada nos algoritmos de gerenciamento do *cluster*, que opera para que não hajam máquinas ligadas sem necessidade.

As simulações foram realizadas em um *cluster*, onde cada nó que o compõe é de uma especificação idêntica, a energia consumida por cada nó sob a mesma carga também é idêntica e todos os trabalhos submetidos ao *cluster* são estritamente intensivos de CPU. Com isso, as latências relacionadas ao acesso à memória, disco e rede não são consideradas.

O estudo fundamentou-se na construção de várias políticas de gestão do *cluster*, baseadas nas estratégias mencionadas anteriormente. As políticas propostas pelos autores são [LAM09]:

- **Gerenciamento do *Cluster*:** essa política é baseada no seguinte esquema de envio de tarefas: todas as máquinas estão sempre ligadas; uma tarefa é atribuída a um único núcleo em um único nó; máquinas são arbitrariamente ordenadas de 0 a N-1, onde N é o número total de nós do *cluster*; quando as tarefas são submetidas, elas são enviadas para a fila de tarefas; tarefas em espera na fila são submetidas a um núcleo do *cluster* usando a ordenação FIFO; as tarefas são submetidas a

menor máquina ordenada (0, ..., N-1) com um núcleo ocioso; todos os núcleos em um único nó devem ser escalados para a mesma frequência, seja a frequência mais alta ou a menor frequência suportada, dependendo de qual política de gerenciamento está em uso no *cluster*. A política de Gerenciamento do *Cluster* é dividida de acordo com o nível de frequência utilizado, em [LAM09]:

- HF (*High Frequency*) - Nesta política, a frequência das CPUs é sempre definida como a maior frequência suportada. Neste caso, todas as máquinas permanecem nesse estado durante toda a duração da simulação.
- LF (*Low Frequency*) - Nesta política, a frequência das CPUs é sempre definida como a menor frequência suportada. Neste caso, todas as máquinas permanecem nesse estado durante toda a duração da simulação.

- **Escalonamento no *Cluster*:** nesta política as máquinas são ligadas e desligadas com base no número de tarefas atualmente em execução no *cluster*, bem como nas tarefas que estão esperando na fila. Inicialmente, todas as máquinas são desligadas. Caso a máquina não utilize escala de frequência, os nós são ativados conforme a necessidade. Já se a máquina utiliza escala de frequência, em primeiro lugar a frequência das máquinas ativas é aumentada, a fim de aumentar o rendimento do *cluster* antes de ligar máquinas adicionais. Sendo que as máquinas são desligadas depois de permanecer em um estado ocioso além de um limite de 300 segundos. A política de Escalonamento do *Cluster* é dividida em [LAM09]:
  - *Scaled to the Highest supported Frequency* (SHF) - Todas as CPUs são escalonadas para a frequência mais alta suportada. Quando os trabalhos são submetidos, eles são atribuídos para os núcleos ociosos do *cluster*.
  - *Scaled to the Lowest supported Frequency* (SLF) - Todas as CPUs são escalonadas para a menor freqüência suportada. Quando os trabalhos são submetidos, eles também são atribuídos para os núcleos ociosos do *cluster*.
  - *Scaled to the lowest Supported freQuency* (SSQ) - Todas as CPUs são inicialmente escalonadas para a menor frequência suportada. Caso hajam tarefas na fila aguardando para serem executadas, as frequências das máquinas atualmente ligadas são aumentadas. Se esse aumento da frequência não for suficiente para atender todas as tarefas em espera na fila, máquinas adicionais são ativadas.
  - *Scaled to the loWest supported freQuency* (SWQ) - Todas as CPUs são inicialmente escalonadas para a menor frequência suportada, permanecendo nessa frequência embora existam núcleos ociosos no *cluster*. Caso o conjunto de nós ativos esteja na capacidade máxima, as frequências das CPUs são aumentadas, de tal forma que a taxa de

transferência do *cluster* é aumentada pelo equivalente a um núcleo adicional (em baixa frequência) para cada tarefa em espera na fila.

- **Escalonamento com Atribuição Inteligente de *Jobs:*** nessa política os autores assumem que a quantidade de tarefas associada a cada posto de trabalho é conhecida no momento da submissão. Dessa maneira, seria possível determinar quais máquinas estão ligadas por mais tempo. A política de Escalonamento com Atribuição Inteligente de *Jobs* está dividida em [LAM09]:
    - SWQI - A política SWQI aperfeiçoa o mecanismo de atribuição de tarefas SWQ. Considerando que a quantidade de "trabalho" exigida por uma tarefa é conhecido no momento da sua submissão, pode-se supor a quantidade de "trabalho" que ainda precisa ser preenchida para cada tarefa no *cluster*. O que, segundo os autores, permite identificar qual a máquina com um núcleo disponível está escalonada por mais tempo. A tarefa seguinte na fila é então atribuída a um núcleo nesta máquina.
    - SWQN - As tarefas são atribuídas às máquinas que estão executando a tarefa mais recentemente apresentada.
    - SWQO - As tarefas são atribuídas às máquinas com a tarefa mais antiga em execução.

Os resultados dos testes realizados por [LAM09] mostraram que:

- HF sempre consome mais energia do que LF, ou seja, um nó é mais eficiente energeticamente quando opera em baixa frequência, mas o tempo de resposta é ruim;
- SWQI e SWQO obtiveram a melhor eficiência energética e desempenho global;
- SHF e SLF necessitam de um número relativamente grande de transições entre ligado e desligado em comparação com o número de transições exigidas pelas políticas que utilizam escala de frequência, como a SWQI e a SWQO. SWQI e SWQO aumentam as freqüências dos processadores no *cluster* antes de energizar nós adicionais, ou seja, máquinas adicionais só são ativadas quando todas as máquinas ativas já foram dimensionadas para uma frequência mais elevada;
- SHF fornece o melhor tempo de resposta, pois as máquinas sempre executam com a frequência máxima suportada. Caso as tarefas começem a acumular na fila, máquinas adicionais são imediatamente ligadas, causando um alívio instantâneo para as tarefas em espera na fila;
- SLF fornece o pior tempo de resposta de todas as políticas que usam escala de frequência. Isso ocorre devido à execução de todos os nós na frequência mínima suportada pela máquina.

Analisando os resultados apresentados, os autores constataram que um nó é energeticamente mais eficiente quando opera em baixa frequência, mas o tempo de resposta é ruim. Sendo que a quantidade de energia consumida aumenta principalmente de acordo com o número de nós ativos no *cluster*. Essa situação pode ser resolvida ao utilizar o dimensionamento automático de nós, que gerencia o tempo que as máquinas permanecem executando baseado na carga de trabalho a que o sistema é submetido.

3.2.5 Galizia et al.

O trabalho de [GAL12] apresenta algumas estratégias de alocação de *jobs* que podem conduzir a economia de energia ao serem aplicadas em um ambiente de grade. As estratégias propostas pelo trabalho visam melhorar a ligação entre os requisitos das aplicações e os recursos da grade, considerando também parâmetros de energia. Para tal, foi criada a métrica de *Energy Congruent Index* (ECI) que classifica os recursos com uma combinação entre desempenho e consumo de energia sendo dependente de um mapeamento entre os recursos da grade e um conjunto de *benchmarks* pré-computados.

O ECI é composto por um vetor de desempenho (P_n) que possui o desempenho de cada um dos nós da grade (valor que é proveniente da execução de *benchmarks* para diversas aplicações distintas) e por um vetor com os valores de consumo de energia de cada nó em *watt* (W_n):

$$ECI_n = \{eci_1, ..., eci_k\}, \text{ onde } eci_j = p_j * w_n, j=1, ..., k$$

Sempre que um novo *job* chega ao escalonador, o mesmo classifica os recursos disponíveis baseado nos valores de eci_h. Dessa forma, o *job* será executado pelo nó livre com o eci_h mínimo. Baseado no ECI, os recursos da grade são classificados em ordem crescente em relação à métrica MFLOPS/W.

Com relação ao consumo de energia, os autores realizaram a comparação da estratégia *Energy-congruent* (EC), com a estratégia *Less Consuming Resource* (LCR) e a *Mega Flops per Watt* (MFW). Na estratégia EC, os *jobs* são atribuídos às máquinas com base no vetor de ECI dos recursos disponíveis. Na LCR, os *jobs* são atribuídos ao recurso que consome menos energia dentre os disponíveis. A estratégia MFW, por sua vez, atribui os *jobs* com base nos valores de MFLOPS/W dos recursos. Os testes foram realizados com dois tipos de *jobs*, um de álgebra linear e o outro de extração de isosuperfície em uma grade composta por três máquinas. Tais máquinas possuíam um desempenho de 1s, 4.9s e 3.4s, e um consumo de energia de 1900W, 1450W e 2200W, respectivamente. Os resultados mostraram que a estratégia LCR é penalizada em desempenho por atribuir os *jobs* à máquina mais lenta, porém que consome menos energia. Já a estratégia MFW se comporta como a EC se a máquina que é melhor em termos de MFLOPS/W (que também é a melhor em termos de desempenho) estiver livre. Entretanto, se ela não estiver a MFW escolherá a máquina que é melhor em termos de

MFLOPS/W, mas que perde em relação ao desempenho, o que causará um aumento no tempo de resposta. Quanto ao consumo de energia, todas as estratégias apresentam aproximadamente os mesmos valores. Segundo os autores, a escolha de uma estratégia dependerá da importância atribuída a ganhar em desempenho ou a economizar significativamente energia.

3.3 Conclusão

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

Tabela 1 – Comparação entre os Trabalhos.

	Ambiente	**Quantidade máquinas**	**Uso DVS**	**Controle temperatura**	**Controle recursos ociosos**	**Ordenação tarefas**	**Base cálculo de consumo**
Ponciano [PON10]	Grade	100	Não	Não	Sim	Não utilizado	Valores pré-definidos fixos para estados de hibernação, sobreaviso e ocioso
Mämmelä [MAM11]	*Cluster*	32	Não	Não	Sim	FIFO, BFF e BBF	Ociosidade dos recursos
Montes [MON12]	Grade	Não especificado	Não	Não	Sim	Não especificado	Valores pré-definidos fixos para os estados das máquinas, independente do tipo de máquina
Wang [WAN11]	*Cluster*	Não especificado	Não	Sim	Não	Não especificado	Temperatura
Kim [KIM07]	*Cluster*	32	Sim	Não	Não	EDF	Voltagem, velocidade relativa e MIPS
Garg [GAR09]	Grade	105	Sim	Não	Não	EDF	Frequência, consumo da CPU, nº de CPUs
Lammie [LAM09]	*Cluster*	75	Sim	Não	Sim	Não especificado	Ociosidade dos recursos e frequência
Galizia [GAL12]	Grade	3	Não	Não	Não	FIFO	Joules
Trabalho Atual	Grade	90	Não	Não	Sim	Tamanho em *flops*	MFW

Como pode ser observado na Tabela 1, a maioria dos trabalhos voltados para grades computacionais ou *clusters* (como [PON10], [MAM11], [MON12] e [LAM09]) utiliza como base para o cálculo de consumo valores relativos ao consumo no estado ocioso dos recursos. Portanto, o cálculo de economia de energia não utiliza os valores relativos a quanto um recurso consome quando está executando tarefas. Apesar do trabalho de [MON12] utilizar o consumo em operação de um recurso, esse valor é fixo (independente do tipo de máquina). Como as máquinas de uma grade são, na grande maioria das vezes, heterogêneas, é possível considerar que esse valor não representa corretamente a realidade.

O trabalho de [WAN11] por sua vez utiliza apenas controle da temperatura para reduzir o consumo de energia em um *cluster*. E os trabalhos de [KIM07] e [GAR09] usam

DVS para obter uma redução no consumo de energia na execução de tarefas que possuem prioridade na ordem de execução (uso do algoritmo de escalonamento EDF).

Nenhum dos trabalhos anteriormente mencionados pode ser repetido a fim de efetuar uma comparação com o trabalho desenvolvido. Uma vez que eles ou utilizam técnicas que não podem ser repetidas devido à falta de um ambiente físico adequado, como as técnicas de DVS e controle de temperatura, ou por utilizarem valores de consumo fixos que não são realísticos para o ambiente de grade que simulamos ([PON10] e [MON12]). Ou ainda, no caso de [MAM11], por utilizar apenas os valores de consumo no estado ocioso dos recursos para calcular o consumo de energia em um ambiente homogêneo de *cluster*.

O trabalho de [GAL12] associa consumo em *watts* e desempenho para tomar decisões de escalonamento e compara a sua solução à métrica MFLOPS/W. Em virtude disso, o algoritmo de escalonamento utilizado por eles é comparado com os algoritmos desenvolvidos neste trabalho. Algumas adaptações foram realizadas por não dispormos dos *benchmarks* utilizados por eles para calcular o desempenho de cada um dos nós da grade que serve como base para o algoritmo de escalonamento tomar decisões. Calcular o desempenho de cada máquina para cada uma das tarefas que irão executar na grade computacional exige muito trabalho. Trabalho que aumenta proporcionalmente em relação à quantidade de máquinas e tarefas, agregando um alto custo computacional e financeiro caso fosse utilizado um ambiente real. Também se utilizou mais máquinas para os testes (90 máquinas), não apenas 3 máquinas como em [GAL12], a fim de construir um ambiente de grade mais realístico.

4 EFICIÊNCIA ENERGÉTICA EM GRADES COMPUTACIONAIS

A fim de explorar soluções de eficiência energética em grades computacionais através de escalonamento de tarefas, desenvolveu-se um módulo para cálculo de consumo de energia e algoritmos de escalonamento energeticamente eficientes dentro da arquitetura da LIBTS no SimGrid. Este capítulo apresenta tais contribuições e traz explicações sobre o seu funcionamento. Inicialmente, descreve-se a nova estrutura da arquitetura, agora com a possibilidade de calcular o consumo de energia na execução de aplicações. Em seguida, apresenta-se o módulo responsável por tal cálculo. Para finalizar, a Seção 4.3 apresenta cada um dos algoritmos de escalonamento desenvolvidos neste trabalho e as alterações realizadas em alguns dos algoritmos mais tradicionais específicos para grades computacionais.

4.1 Arquitetura de Escalonamento

A estrutura da arquitetura de escalonamento baseia-se na da apresentada na Seção 2.4, com a inclusão do módulo de cálculo de consumo de energia (explicação na próxima seção) e dos algoritmos de escalonamento energeticamente eficientes implementados neste trabalho. Os algoritmos são denominados como:

- *Low Energy Consumption Scheduling Algorithm* (LECSA);
- *Low Energy Consumption Scheduling Algorithm Version* 2 (LECSA2);
- *Low Energy Consumption Scheduling Algorithm Version* 3 (LECSA3);
- *Dynamic Low Energy Consumption Scheduling Algorithm* (DLECSA);
- *Dynamic Low Energy Consumption Scheduling Algorithm Version* 2 (DLECSA2);
- *Dynamic Low Energy Consumption Scheduling Algorithm Version* 3 (DLECSA3).

Os algoritmos acima mencionados podem ser classificados segundo a taxonomia hierárquica para algoritmos de escalonamento de *Casavant* e *Kuhl*, conforme retratado na Seção 2.2.1. De acordo com essa taxonomia o LECSA, o LECSA2 e o LECSA3 são classificados como: global, estático, sub-ótimo, heurístico, não preemptivo e direcionados para a execução em sistemas heterogêneos. Já o DLECSA, o DLECSA2 e o DLECSA3 são classificados como: global, dinâmico, não-distribuído, não preemptivo que operam em sistemas heterogêneos.

Também foram realizadas alterações nos algoritmos WQ, *Sufferage*, *XSufferage* e DFPLTF para que eles tomassem decisões de escalonamento baseados em energia e não somente em desempenho. Já o algoritmo ECI [GAL12] foi implementado para fins de comparação com algoritmos de escalonamento diretamente voltados à redução de consumo de energia. A arquitetura pode ser observada na Figura 9. As partes em destaque representam os algoritmos de escalonamento voltados para eficiência

energética e o módulo de cálculo de consumo desenvolvidos neste trabalho. Como pode ser observado na figura, os algoritmos são divididos em duas classes: uma voltada para desempenho, contendo 5 algoritmos, e outra voltada para eficiência energética, contendo 11 algoritmos.

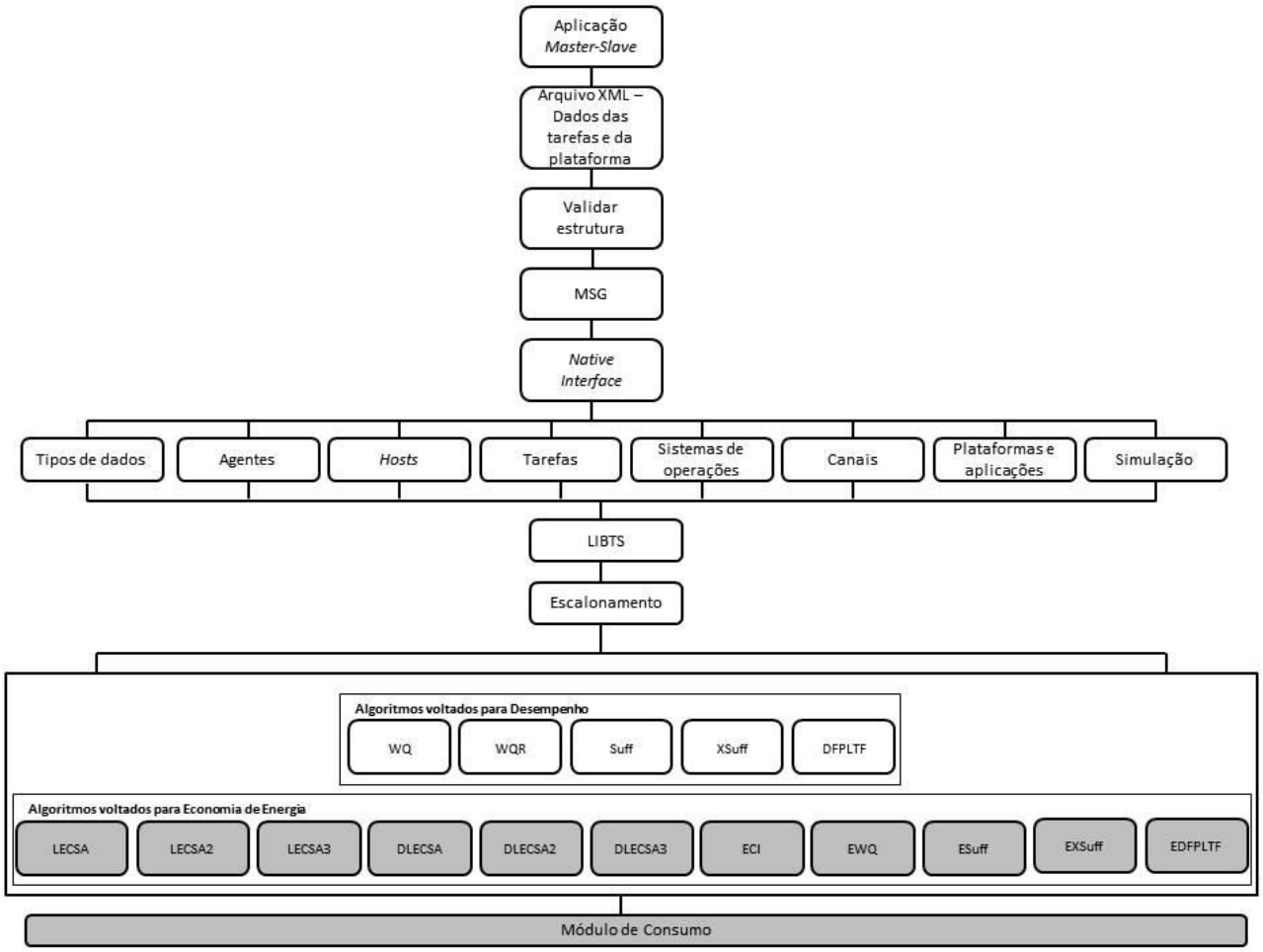

Figura 9 – Arquitetura de escalonamento.

4.2 Módulo de Cálculo de Consumo

Com o objetivo de estimar o consumo de energia na execução de uma aplicação e, posteriormente, saber quanta energia foi gasta por cada algoritmo de escalonamento, desenvolveu-se o módulo de consumo. Tal módulo calcula o consumo de energia baseado no consumo do *host* para executar determinada tarefa, no consumo do *host* no estado ocioso e no consumo da transferência de uma tarefa. Ou seja, são consideradas mais informações para o cálculo de consumo de energia do que em alguns dos trabalhos relacionados apresentados no capítulo 3, que só consideram o consumo da máquina no estado ocioso. Este mecanismo foi implementado dentro do *masterslave.c* da LIBTS, para que fosse possível calcular: o consumo de energia de cada tarefa, de cada máquina (*host*) e o consumo total do algoritmo de escalonamento utilizado na execução da aplicação. O módulo requer cinco informações para funcionar corretamente: (i) o tamanho da tarefa, (ii) o consumo em *flops/watt* de um *host*, (iii) o tempo que um *host* permanece

ocioso, (iv) o consumo de energia de um *host* no estado ocioso e (v) o consumo de energia dos *links* que conectam o *host* mestre (responsável por enviar a tarefa para o *host* a qual foi escalonada) ao *host* escravo. Este módulo pode ser adaptado para qualquer ambiente de simulação com o mesmo modelo de energia e as medidas de consumo.

A Equação (1) calcula o consumo de energia de cada tarefa. Onde, *taskSize* é o tamanho em *flops* da tarefa e o *consumption* é a quantidade de *flops* que o *host* (no qual a tarefa foi alocada) consegue executar por *watt* de energia consumido. Esta medição é utilizada em diversas pesquisas, incluindo a lista Top500 [TOP11].

$$ecTask = taskSize / consumption \quad (1)$$

O consumo de energia da transferência de uma tarefa do *host* mestre para o *host* escravo é dado pela Equação (2). Para esse cálculo são usadas as informações do tamanho da tarefa em *bits* (*taskSizeBits*), do tempo que essa tarefa levou para ser transferida em segundos (*timeTransf*) e do consumo do *link* de *conexão* (*consumptionLink)* dado em *watts/bps* (quantidade de *watts* consumidos para transferir um *bit* por segundo).

$$ecTaskNetwork = (taskSizeBits / timeTransf) * consumptionLink \quad (2)$$

A Equação (3) calcula o consumo de energia de cada *host.* É utilizada a soma do consumo de cada tarefa que foi executada por ele ($ecTask_i$), somado ao consumo de cada *host* no estado ocioso e ao consumo da transferência de cada tarefa executada pelo *host*. O valor do consumo ocioso é calculado multiplicando-se o tempo que o *host* permanece no estado ocioso *(timeIdle*) pelo consumo ocioso desse *host* (*ecIdle*). Sempre que um *host* termina de executar uma tarefa começa a contar o seu tempo no estado ocioso, quando ele recebe a próxima tarefa ele sai do estado ocioso para o estado ativo, e assim sucessivamente. Dessa forma, um acumulador calcula o tempo total que cada *host* permanece no estado ocioso.

$$ecHost_i = (\Sigma ecTask_i) + (timeIdle * ecIdle) + (\Sigma ecTaskNetwork_i) \quad (3)$$

Já o consumo total do algoritmo, é a soma do consumo de cada *host*, dado pela Equação (4):

$$ecTotal = \Sigma ecHost_i \quad (4)$$

Com o uso dessas equações, é possível estimar a quantidade de energia consumida na execução de uma aplicação de acordo com o algoritmo de escalonamento utilizado.

4.3 Algoritmos de Escalonamento Energeticamente Eficientes

Nesta seção, são apresentados os algoritmos energeticamente eficientes desenvolvidos neste trabalho. Estes estão divididos em duas classes: a classe estática e a dinâmica. A classe estática contém os seguintes algoritmos: LECSA (Seção 4.3.1), LECSA2 (Seção 4.3.2) e LECSA3 (Seção 4.3.3). A classe dinâmica contém: DLECSA

(Seção 4.3.4), o DLECSA2 (Seção 4.3.5) e o DLECSA3 (Seção 4.3.6). As próximas seções explicam em detalhes o funcionamento de cada um dos algoritmos, além de mostrá-los através de pseudo-códigos.

4.3.1 *Low Energy Consumption Scheduling Algorithm* (LECSA)

O LECSA foi o primeiro algoritmo desenvolvido, toda a sua lógica é direcionada para reduzir o consumo de energia. A seguir é detalhada a estrutura do LECSA e na Figura 10 apresenta-se o seu pseudo-código.

1. Inicialmente (linha 2 do pseudo-código da Figura 10), as tarefas são ordenadas em ordem decrescente de acordo com o seu *taskSize*;
2. Em seguida (linha 4), os *hosts* são ordenados em ordem decrescente, de acordo com a sua eficiência energética (*flops/watt*);
3. De acordo com a variação no tamanho das tarefas que compõe uma aplicação (a variação do tamanho das tarefas é descrita na Seção 5.2, de 0 a 100%), é escolhida a quantidade de *hosts* que será utilizada na execução (linhas 6-10). Caso o número de tarefas for maior que a quantidade de *hosts,* então, a quantidade de máquinas utilizadas é baseada no percentual de variação de heterogeneidade das tarefas pela quantidade de *hosts* (linha 7). Por exemplo, com 90 máquinas tendo uma variação de 50% no tamanho das tarefas, serão utilizadas 45 máquinas (caso a variação for de 0%, por padrão, utilizam-se 25% das máquinas). Caso o número de tarefas for menor que a quantidade de *hosts,* então, a quantidade de máquinas utilizadas é baseada na variação pelo número de tarefas (linha 9). Por exemplo, com 90 máquinas tendo variação de 50% no tamanho das tarefas e tendo 30 tarefas para escalonar, serão utilizadas 15 máquinas*;*
4. As máquinas que não serão utilizadas são desligadas para evitar que fiquem ociosas consumindo energia desnecessariamente (linhas 12-15);
5. O próximo passo consiste em dividir o número total de tarefas pelo número total de *hosts*, para que cada *host* receba a mesma quantidade de tarefas para executar (linha 17);
6. Depois disso, é percorrido cada *host* e enviada uma tarefa por vez até que todas as tarefas tenham sido enviadas (linhas 19-27). Para enviar as tarefas existem controles que garantem que as com maior *taskSize* sejam enviadas para os *hosts* com melhor eficiência energética. Na linha 21, é pré-definida a próxima tarefa da fila a ser enviada. Na linha 22, é percorrida a lista de *hosts* e na sequência (linha 23) envia-se a tarefa. Em seguida, é verificada qual deve ser a próxima tarefa a ser enviada. Para isso, é incrementado o índice da tarefa a ser enviada com o valor da quantidade de tarefas por *host*;

7. Após o término de todas as tarefas, os *hosts* que foram utilizados na execução são finalizados (linhas 29-32).

Entrada: *tasks*[n]: lista de tarefas para alocar; *task_count*: quantidade de tarefas; *slaves[n]*: lista de *hosts*; *slave_count*: quantidade de *hosts*; *variation*: variação da hgeterogeneidade das tarefas.
Variáveis: *num_hosts*: quantidade de *hosts* que serão utilizados; *num_tasks_host*: quantidade de tarefas que devem ser escalonadas em cada *host*; *task_send*: tarefa que deverá ser enviada na rodada

```
1.  // ordena as tarefas pelo taskSize
2.  tasks[task_count] ← sortTasks(tasks[task_count])
3.  // ordena os hosts por flops/watt
4.  slaves[slave_count] ← sortHostsConsumption(slaves[slave_count])
5.  // verifica através da variação na heterogeneidade das tarefas a quantidade de hosts que serão utilizados
6.  IF task_count > slave_count THEN
7.     num_hosts ← (slave_count * variation) / 100
8.  ELSE
9.     num_hosts ← (task_count * variation) / 100
10. ENDIF
11. // finaliza os hosts que não serão usados para não consumir energia desnecessariamente
12. DO
13.    FOR i ← num_hosts; i < slave_count IN slaves[i]
14.       finalizeHost(slaves[i])
15. ENDDO
16. // verifica quantas tarefas serão alocadas em cada host
17. num_tasks_host ← task_count / num_hosts
18. // percorre toda lista de tarefas e aloca uma tarefa de cada host por vez
19. DO
20.    FOR i ← 0; i < tasks_count IN tasks[i]
21.       task_send ← i
22.       FOR k ← 0; k < num_hosts IN slaves[k]
23.          sendTask(tasks[task_send], slave[k])
24.          task_send ← task_send + num_tasks_host
25.       END
26.    END
27. ENDDO
28. // finaliza os hosts que foram utilizados
29. DO
30.    FOR i ← 0; i < num_hosts IN slaves[i]
31.       finalizeHost(slaves[i])
32. ENDDO
```

Figura 10 – Pseudo-código do LECSA.

Dessa forma, garante-se o envio das maiores tarefas para os *hosts* mais econômicos e tarefas menores para os *hosts* com menor eficiência energética.

4.3.2 *Low Energy Consumption Scheduling Algorithm Version 2* (LECSA2)

O LECSA2 foi desenvolvido com o objetivo de melhorar o desempenho do LECSA. Após a realização de alguns testes verificou-se que, em alguns casos testados, a perda de desempenho mostrava-se muito expressiva, apesar da melhoria na eficiência energética em comparação com os demais algoritmos tradicionais específicos para grades. Em virtude disso, foi acrescentada ao código original do LECSA, uma função que ordena os *hosts*, que serão utilizados no escalonamento das tarefas, pelo seu poder computacional. A Figura 11 apresenta o pseudo-código do LECSA2. A estrutura do LECSA2 é idêntica a do LECSA nos passos 1-4, mas o passo 5 do LECSA2 consiste na

ordenação das máquinas que serão utilizadas no escalonamento pelo seu poder computacional, em ordem decrescente (linha 17 do pseudo-código da Figura 11). Os passos 5-6 do LECSA são repetidos no LECSA2, depois da função de ordenação de *hosts* pelo desempenho (linha 17), descrita anteriormente. Dessa forma, garante-se que as tarefas sejam enviadas para as máquinas com melhor desempenho dentre as com melhor eficiência energética.

Entrada: *tasks[n]*: lista de tarefas para alocar; *task_count*: quantidade de tarefas; *slaves[n]*: lista de *hosts*; *slave_count*: quantidade de *hosts*; *variation*: variação da heterogeneidade das tarefas.
Variáveis: *num_hosts*: quantidade de *hosts* que serão utilizados; *num_tasks_host*: quantidade de tarefas que devem ser escalonadas em cada *host*; *task_send*: tarefa que deverá ser enviada na rodada

```
1.  // ordena as tarefas pelo taskSize
2.  tasks[task_count] ← sortTarefas(tasks[task_count])
3.  // ordena os hosts por flops/watt
4.  slaves[slave_count] ← sortHostsConsumption(slaves[slave_count])
5.  // verifica através da variação na heterogeneidade das tarefas a quantidade de hosts que serão utilizados
6.  IF task_count > slave_count THEN
7.     num_hosts ← (slave_count * variation) / 100
8.  ELSE
9.     num_hosts ← (task_count * variation) / 100
10. ENDIF
11. // finaliza os hosts que não serão usados para não consumir energia desnecessariamente
12. DO
13.    FOR i ← num_hosts; i < slave_count IN slaves[i]
14.       finalizeHost(slaves[i])
15. ENDDO
16. // ordena os hosts por flops (poder computacional)
17. slaves[slave_count] ← sortHostsPerformance(slaves[slave_count])
18. // verifica quantas tarefas serão alocadas em cada host
19. num_tasks_host ← task_count / num_hosts
20. // percorre toda lista de tarefas e aloca uma tarefa de cada host por vez
21. DO
22.    FOR i ← 0; i < tasks_count IN tasks[i]
23.       task_send ← i
24.       FOR k ← 0; k < num_hosts IN slaves[k]
25.          sendTask(tasks[task_send], slave[k])
26.          task_send ← task_send + num_tasks_host
27.       END
28.    END
29. ENDDO
30. // finaliza os hosts que foram utilizados
31. DO
32.    FOR i ← 0; i < num_hosts IN slaves[i]
33.       finalizeHost(slaves[i])
34. ENDDO
```

Figura 11 – Pseudo-código do LECSA2.

4.3.3 *Low Energy Consumption Scheduling Algorithm Version 3* (LECSA3)

Como a alteração no LECSA2 não atingiu em alguns casos testados o ganho de desempenho esperado, implementou-se o LECSA3 cujas alterações em relação ao LECSA2 podem ser observadas pela Figura 12. Entretanto, pelo fato do LECSA3 ser mais direcionado para obter ganho de desempenho do que o LECSA e o LECSA2, ele

apresenta uma redução na eficiência energética, em comparação com os algoritmos citados.

As mudanças estão no momento de ordenação dos *hosts* em relação ao desempenho e a eficiência energética. O passo 1 do LECSA2 permanece igual, mas o passo 2 muda. No passo 2, agora os *hosts* são ordenados em ordem decrescente pelo seu poder computacional (linha 4 do pseudo-código da Figura 12). Em seguida, os passos 3 e 4 do LECSA2 são executados. O passo 5 do LECSA3 também sofre alteração em relação ao do LECSA2. Esse passo, agora consiste na ordenação das máquinas que serão utilizadas no escalonamento pela sua eficiência energética, em ordem decrescente (linha 17). Os passos 5-6 do LECSA são repetidos no LECSA3, depois da função de ordenação de *hosts* pela eficiência energética descrita anteriormente. O objetivo das modificações realizadas é garantir que as tarefas sejam enviadas para as máquinas com melhor eficiência energética dentre as com melhor desempenho.

Entrada: *tasks[n]*: lista de tarefas para alocar; *task_count*: quantidade de tarefas; *slaves[n]*: lista de *hosts*; *slave_count*: quantidade de *hosts; variation*: variação da heterogeneidade das tarefas.
Variáveis: num_hosts: quantidade de *hosts* que serão utilizados; *num_tasks_host*: quantidade de tarefas que devem ser escalonadas em cada *host*; *task_send*: tarefa que deverá ser enviada na rodada

```
1.  // ordena as tarefas pelo taskSize
2.  tasks[task_count] ← sortTasks(tasks[task_count])
3.  // ordena os hosts por flops (poder computacional)
4.  slaves[slave_count] ← sortHostsPerformance(slaves[slave_count])
5.  // verifica através da variação na heterogeneidade das tarefas a quantidade de hosts que serão utilizados
6.  IF task_count > slave_count THEN
7.     num_hosts ← (slave_count * variation) / 100
8.  ELSE
9.     num_hosts ← (task_count * variation) / 100
10. ENDIF
11. // finaliza os hosts que não serão usados para não consumir energia desnecessariamente
12. DO
13.    FOR i ← num_hosts; i < slave_count IN slaves[i]
14.       finalizeHost(slaves[i])
15. ENDDO
16. // ordena os hosts por flops/watt
17. slaves[slave_count] ← sortHostsConsumption(slaves[slave_count])
18. // verifica quantas tarefas serão alocadas em cada host
19. num_tasks_host ← task_count / num_hosts
20. // percorre toda lista de tarefas e aloca uma tarefa de cada host por vez
21. DO
22.    FOR i ← 0; i < tasks_count IN tasks[i]
23.       task_send ← i
24.       FOR k ← 0; k < num_hosts IN slaves[k]
25.          sendTask(tasks[task_send], slave[k])
26.          task_send ← task_send + num_tasks_host
27.       END
28.    END
29. ENDDO
30. // finaliza os hosts que foram utilizados
31. DO
32.    FOR i ← 0; i < num_host IN slaves[i]
33.       finalizeHost(slaves[i])
34. ENDDO
```

Figura 12 – Pseudo-código do LECSA3.

4.3.4 *Dynamic Low Energy Consumption Scheduling Algorithm* (DLECSA)

O objetivo do desenvolvimento do DLECSA é criar uma versão dinâmica do LECSA. Devido aos outros algoritmos de escalonamento, específicos para grades, testados serem dinâmicos. Sua dinamicidade é proveniente do fato de um *host* receber a próxima tarefa da fila assim que finalizar a execução da tarefa atual, o mesmo não ocorre nas versões estáticas do LECSA. Nas versões estáticas, as tarefas que irão para cada *host* são pré-definidas antes do início da execução da aplicação. Sendo que cada *host* recebe a mesma quantidade de tarefas o que não ocorre nas versões dinâmicas. A Figura 13 apresenta o pseudo-código do DLECSA. A seguir descreve-se a estrutura do algoritmo.

Entrada: *tasks[n]*: lista de tarefas para alocar; *task_count*: quantidade de tarefas; *slaves[n]*: lista de *hosts*; *slave_count*: quantidade de *hosts*; *variation*: variação da hgeterogeneidade das tarefas.
Variáveis: *num_hosts*: quantidade de *hosts* que serão utilizados; *num_tasks_host*: quantidade de tarefas que devem ser escalonadas em cada *host*

```
1.  // ordena os hosts por flops/watt
2.  slaves[slave_count] ← sortHostsConsumption(slaves[slave_count])
3.  // verifica através da variação na heterogeneidade das tarefas a quantidade de hosts que serão utilizados
4.  IF task_count > slave_count THEN
5.     num _hosts ← (slave_count * variation) / 100
6.  ELSE
7.     num_hosts ← (task_count * variation) / 100
8.  ENDIF
9.  // finaliza os hosts que não serão usados para não consumir energia desnecessariamente
10. DO
11.    FOR i ← num_hosts; i < slave_count IN slaves[i]
12.       finalizeHost(slaves[i])
13. ENDDO
14. // percorre toda lista de tarefas e aloca a próxima tarefa da fila no primeiro host que ficar livre
15. DO
16.    FOR i ← 0; i < tasks_count IN tasks[i]
17.       FOR k ← 0; k < num_hosts IN slaves[k]
18.          // se o slave estiver livre, envia a tarefa. Caso contrário, verifica se o próximo está livre
19.          IF freeSlave(slaves[k]) THEN
20.             sendTask(tasks[i], slave[k])
21.          ENDIF
22.       END
23.    END
24. ENDDO
25. // finaliza os hosts que foram utilizados
26. DO
27.    FOR i ← 0; i < num_hosts IN slaves[i]
28.       finalizeHost(slaves[i])
29. ENDDO
```

Figura 13 – Pseudo-código do DLECSA.

1. Inicialmente, os *hosts* são ordenados em ordem decrescente (linha 2 do pseudo-código da Figura 13), de acordo com a sua eficiência energética (*flops/watt*);
2. Este passo é idêntico ao passo 3 do algoritmo LECSA, apresentado na Seção 4.3.1 (linhas 4-8);
3. Este passo é idêntico ao passo 4 do algoritmo LECSA, apresentado na Seção 4.3.1 (linhas 10-13);

4. A primeira tarefa da fila é escalonada no primeiro *host* livre encontrado (os *hosts* estão ordenados pela eficiência energética). Dessa forma, os *hosts* mais econômicos receberão as tarefas, desde que estejam livres. Por exemplo, se o primeiro *host* (mais econômico) estiver ocupado, então, o segundo *host* (menos econômico) receberá a tarefa (linhas 14-24);

5. Após o término de todas as tarefas, os *hosts* que foram utilizados na execução são finalizados (linhas 26-29).

4.3.5 *Dynamic Low Energy Consumption Scheduling Algorithm Version 2* (DLECSA2)

Da mesma maneira que no LECSA, foram criadas duas novas versões do DLECSA em prol de alcançar melhorias em termos de desempenho. O funcionamento do DLECSA2 é idêntico ao do DLECSA dos passos 1-3 (linhas 2-13 do pseudo-código da Figura 14). Já no passo 4, a primeira tarefa da fila é escalonada para o primeiro *host* livre encontrado com maior poder computacional (linhas 15-24). A meta é garantir que o *host* com melhor desempenho dentre os livres receba a próxima tarefa da fila.

Entrada: *tasks[n]*: lista de tarefas para alocar; *task_count*: quantidade de tarefas; *slaves[n]*: lista de *hosts*; *slave_count*: quantidade de *hosts*; *variation*: variação da hgeterogeneidade das tarefas.
Variáveis: *num_hosts*: quantidade de *hosts* que serão utilizados; *num_tasks_host*: quantidade de tarefas que devem ser escalonadas em cada *host*

```
1.  // ordena os hosts por flops/watt
2.  slaves[slave_count] ← sortHostsConsumption(slaves[slave_count])
3.  // verifica através da variação na heterogeneidade das tarefas a quantidade de hosts que serão utilizados
4.  IF task_count > slave_count THEN
5.     num_hosts ← (slave_count * variation) / 100
6.  ELSE
7.     num_hosts ← (task_count * variation) / 100
8.  ENDIF
9.  // finaliza os hosts que não serão usados para não consumir energia desnecessariamente
10. DO
11.    FOR i ← num_hosts; i < slave_count IN slaves[i]
12.       finalizeHost(slaves[i])
13. ENDDO
14. // percorre toda lista de tarefas e aloca a próxima tarefa da fila no host livre com melhor desempenho
15. DO
16.    FOR i ← 0; i < tasks_count IN tasks[i]
17.       FOR k ← 0; k < num_hosts IN slaves[k]
18.          // se o slave estiver livre, envia a tarefa. Caso contrário, verifica se o próximo está livre
19.          IF freeSlave(slaves[k]) AND bestHostPerformance(slaves[k]) THEN
20.             sendTask(tasks[i], slave[k])
21.          ENDIF
22.       END
23.    END
24. ENDDO
25. // finaliza os hosts que foram utilizados
26. DO
27.    FOR i ← 0; i < num_hosts IN slaves[i]
28.       finalizeHost(slaves[i])
29. ENDDO
```

Figura 14 – Pseudo-código do DLECSA2.

O último passo ocorre após o término de todas as tarefas. Nesse passo, ocorre a finalização das máquinas que foram utilizados na execução (linhas 26-29). Como pode ser observado na Figura 14, a lógica do DLECSA2 é similar a do LECSA2, exceto pelo fato do primeiro ser dinâmico e o segundo estático, ou seja, a diferença está na forma de realizar o envio das tarefas.

4.3.6 *Dynamic Low Energy Consumption Scheduling Algorithm Version 3* (DLECSA3)

No DLECSA3, o primeiro passo consiste na ordenação dos *hosts* pelo poder computacional, em ordem decrescente (linha 2 do pseudo-código da Figura 15). Os próximos passos são idênticos aos passos 2 e 3 do DLECSA (linhas 4-13). A etapa seguinte realiza o envio da primeira tarefa da fila ao primeiro *host* livre encontrado com maior eficiência energética (linhas 15-24). A meta é garantir que o *host* mais eficiente energeticamente dentre os livres, receba a próxima tarefa da fila. O último passo é idêntico ao passo 5 do DLECSA (Seção 4.3.4) (linhas 26-29). Na Figura 15, pode ser observado o pseudo-código do DLECSA3.

Entrada: *tasks[n]*: lista de tarefas para alocar; *task_count*: quantidade de tarefas; *slaves[n]*: lista de *hosts*; *slave_count*: quantidade de *hosts*; *variation*: variação da hgeterogeneidade das tarefas.
Variáveis: *num_hosts*: quantidade de *hosts* que serão utilizados; *num_tasks_host*: quantidade de tarefas que devem ser escalonadas em cada *host*

```
1.  // ordena os hosts pelo poder computacional
2.  slaves[slave_count] ← sortHostsPerformance(slaves[slave_count])
3.  // verifica através da variação na heterogeneidade das tarefas a quantidade de hosts que serão utilizados
4.  IF task_count > slave_count THEN
5.     num _hosts ← (slave_count * variation) / 100
6.  ELSE
7.     num_hosts ← (task_count * variation) / 100
8.  ENDIF
9.  // finaliza os hosts que não serão usados para não consumir energia desnecessariamente
10. DO
11.    FOR i ← num_hosts; i < slave_count IN slaves[i]
12.       finalizeHost(slaves[i])
13. ENDDO
14. // percorre toda lista de tarefas e aloca a próxima tarefa da fila no host livre mais eficiente energeticamente
15. DO
16.    FOR i ← 0; i < tasks_count IN tasks[i]
17.       FOR k ← 0; k < num_hosts IN slaves[k]
18.          // se o slave estiver livre, envia a tarefa. Caso contrário, verifica se o próximo está livre
19.          IF freeSlave(slaves[k]) AND bestHostConsumption(slaves[k]) THEN
20.             sendTask(tasks[i], slave[k])
21.          ENDIF
22.       END
23.    END
24. ENDDO
25. // finaliza os hosts que foram utilizados
26. DO
27.    FOR i ← 0; i < num_hosts IN slaves[i]
28.       finalizeHost(slaves[i])
29. ENDDO
```

Figura 15 – Pseudo-código do DLECSA3.

4.3.7 *Energy Workqueue* (EWQ)

O comportamento do algoritmo EWQ é similar ao do WQ, exceto pelo fato do EWQ ser voltado para economia de energia. Dessa forma, inicialmente é realizada a ordenação dos *hosts* pela sua eficiência energética. Em seguida, as tarefas são enviadas primeiramente para os *hosts* mais econômicos. O objetivo é que um maior número de tarefas seja atribuído para máquinas com menor consumo de energia. Mas, caso essas máquinas mais eficientes energeticamente, sejam ruins em termos de desempenho, pode ocorrer das tarefas serem enviadas para máquinas com menor eficiência energética. Isso ocorre, pois a próxima tarefa da fila é enviada para o *host* que tornar-se disponível primeiro.

4.3.8 Energy Sufferage (ESuff)

A lógica do algoritmo ESuff é similar a da sua versão original. Entretanto, o objetivo não é melhorar o desempenho com no *Sufferage*, mas sim melhorar a eficiência energética. Para tal, ao invés de calcular o quanto cada tarefa poderia ser prejudicada se não for escalonada no processador que a execute de forma mais eficiente em termos de desempenho, é calculado o quanto cada tarefa poderia ser prejudicada se não fosse escalonada na máquina que a execute com maior eficiência energética. Dessa forma, uma tarefa sempre será executada pelo *host* que apresentar o menor consumo de energia dentre todos os *hosts* da plataforma.

4.3.9 *Energy XSufferage* (EXSuff)

A principal diferença entre o ESuff e o EXSuff é o método usado para calcular o valor *sufferage*. O EXSuff considera a transferência dos dados de entrada da tarefa durante o cálculo dos tempos de execução. Na Seção 2.2.2, é explicado o cálculo do valor *sufferage*. Lembrando que foram feitas alterações nesse cálculo para que ele fosse voltado para eficiência energética, como dito na seção anterior.

4.3.10 *Energy Dynamic Fastest Processor to Largest Task First* (EDFPLTF)

No algoritmo EDFPLTF, inicialmente as tarefas são ordenadas por tamanho em ordem decrescente da mesma maneira que é feito na sua versão original (Seção 2.2.2). A diferença do EDFPLTF em relação ao DFPLTF fundamenta-se na meta de que a maior tarefa seja a primeira a ser alocada para o *host* que possuir o menor consumo de energia (no DFPLTF a maior tarefa é a primeira a ser alocada para o *host* que provê o menor tempo de execução).

5 RESULTADOS OBTIDOS

O objetivo deste capítulo é, além de avaliar o mecanismo de cálculo de consumo, analisar também o desempenho e o consumo de energia dos algoritmos desenvolvidos neste trabalho em relação a outros algoritmos de escalonamento para grades computacionais já existentes. Para tal, utilizaram-se diversas configurações de máquinas visando aproximar-se de casos reais juntamente com distintas configurações de filas de tarefas.

Entre os itens analisados no decorrer deste capítulo, encontram-se: avaliação da influência da heterogeneidade das máquinas (diferentes velocidades), da heterogeneidade das tarefas (diferentes tamanhos) e da granularidade das aplicações (variações na relação máquinas por tarefas) no comportamento dos algoritmos desenvolvidos. Nas seções que seguem, apresenta-se o ambiente de simulação e os cenários de testes utilizados, e por fim a análise dos resultados obtidos.

5.1 Ambiente de Simulação

Esta seção apresenta o ambiente de simulação utilizado para os testes da abordagem de gerenciamento de eficiência energética em grades computacionais desenvolvida por este trabalho. O ideal seria realizar os experimentos em um ambiente real de grade, pois os resultados seriam mais confiáveis por representarem o comportamento do sistema real. No entanto, na computação em grade isso é muito raro, pois a construção e manutenção de uma infraestrutura de grade é difícil e financeiramente cara. Neste cenário, simular o comportamento de ambientes reais de grade torna-se uma alternativa viável, pois não há a necessidade de alocar uma infraestrutura real para realizar experimentos. Além disso, com o uso de simulação é possível realizar diversos experimentos e combinar diferentes cenários utilizando muito menos tempo.

As ferramentas escolhidas para definir o ambiente experimental foram o SimGrid [SIM12] e a LIBTS [FRA11]. A escolha deste sistema de simulação deve-se à sua compatibilidade com o nosso modelo de eficiência energética, onde o poder computacional das máquinas é representado por *flops* e o consumo de energia pode ser calculado em *flops/watt*. Nos testes, foram utilizados os algoritmos WQ, WQR, Suff, XSuff, e DFPLTF, uma vez que esses algoritmos de escalonamento são específicos para grades. A grade simulada é composta por 90 máquinas. Como pode ser observado na Figura 9 (Seção 4.1), o módulo de cálculo de consumo foi integrado à arquitetura, a fim de recolher a energia gasta por cada algoritmo durante a execução das aplicações. Além disso, os algoritmos LECSA, LECSA2, LECSA3, DLECSA, DLECSA2, DLECSA3, EWQ, ESuff, EXSuff, ECI e EDFPLTF foram adicionados ao conjunto de algoritmos de escalonamento implementados na LIBTS.

5.2 Cenários de Testes

Nesta seção, descreve-se o ambiente utilizado para construir e executar os testes dos algoritmos desenvolvidos. Bem como, apresentam-se as tarefas utilizadas nos testes realizados, para a avaliação do comportamento, desempenho e consumo de energia dos novos algoritmos de escalonamento em comparação com os já existentes. Inicialmente, explica-se a granularidade de aplicações utilizadas nos experimentos. Em seguida, a heterogeneidade das máquinas e das tarefas e, finalmente, a configuração do consumo das máquinas.

Granularidade das Aplicações

Definiu-se a granularidade de aplicações, do tipo BoT, com base no trabalho de [SIL03] que estabelece a base para a construção de cenários de teste para experimentos de algoritmos de escalonamento em grade. Foram construídos três grupos de tarefas compostos por aplicações divididas em diferentes quantidades de tarefas: 29, 144 e 720. Para cada grupo, há um tamanho médio de tarefa (ver Tabela 2). O objetivo é configurar diferentes cenários, testando situações em que o número de tarefas é muito menor do que o número de máquinas, e em que o número de tarefas é muito maior do que o número de máquinas.

Tabela 2 – Granularidade das Aplicações.

Quantidade de tarefas	Máquinas/tarefas	Tamanho médio das tarefas (*Tflops*)
720	0.105	10
144	0.605	50
29	3.103	250

Nos experimentos as aplicações possuem o mesmo tamanho. Dessa forma, no caso em que o tamanho médio das tarefas é 10 *Tflops* (grão fino) haverá muito mais tarefas do que quando o tamanho médio das tarefas for de 250 *T*flops (grão grosso).

Heterogeneidade das Máquinas

Utilizaram-se três plataformas de máquinas, com diferentes níveis de heterogeneidade entre suas máquinas a fim de reproduzir o mais próximo possível a configuração real das grades computacionais. A velocidade das máquinas é definida de acordo com uma distribuição uniforme, mantendo a média de velocidade de todas as máquinas da grade em aproximadamente 12,36 *Gflops*. Os níveis de heterogeneidade das máquinas definidos são (Apêndice A):

- **Plataforma 1:** baixa heterogeneidade, a velocidade das máquinas varia de acordo com uma distribuição uniforme de 11,23 *Gflops* até 13,40 *Gflops*;
- **Plataforma 2:** média heterogeneidade, a velocidade das máquinas varia de acordo com uma distribuição uniforme de 7,66 *Gflops* até 14,31 *Gflops*;

- **Plataforma 3:** alta heterogeneidade, a velocidade das máquinas varia de acordo com uma distribuição uniforme de 4,71 *Gflops* até 20,31 *Gflops*.

Heterogeneidade das Tarefas

As tarefas podem ter tamanho médio de 10 *Tflops*, 50 *Tflops* e 250 *Tflops*. Sendo que, o tamanho das tarefas varia em cada grupo, mas a média de tamanho permanece a mesma do grupo. A variação no tamanho das tarefas pode levá-las a serem homogêneas (0% de variação) a completamente heterogêneas (100% de variação). Os intervalos utilizados foram de 0%, 25%, 50%, 75% e 100% de variação no tamanho médio das tarefas do grupo. Por exemplo, para um intervalo de 25% de variação, os tamanhos das tarefas podem apresentar valores de 12.5% abaixo e 12.5% acima do tamanho médio (10 *Tflops*; 50 *Tflops* ou 250 *Tflops*). Caso o tamanho médio for de 250 *Tflops*, o tamanho de cada tarefa variará de 218,75 *Tflops* a 281,25 *Tflops*, com o intervalo de 25% de variação. A aplicação recebe tarefas de acordo com uma distribuição, até que a soma de seus tamanhos alcance o tamanho da aplicação que é de no máximo 7250 *Tflops.*

Configuração de Consumo das Máquinas

Encontrar o poder em *flops* e consumo em *flops/watt* de máquinas "*off-the-shelf*" não é uma tarefa trivial. Por essa razão, os valores de consumo de energia em *flops/watt* das máquinas utilizadas na simulação, foram definidos conforme a configuração de máquinas reais extraídos dos trabalhos de [SHA06] e [GAL12]. A Tabela 3 apresenta na primeira coluna a descrição de cada máquina, seguida pelos valores do poder computacional, consumo de energia em *flops/watt* e em *watt*, baseados nos trabalhos de [SHA06] e [GAL12] e consumo de energia no estado de ociosidade, baseado no [ENE12]. As máquinas utilizadas como base para a simulação foram escolhidas de acordo com as suas características, tentando manter o ambiente simulado mais realista. Objetivando construir um ambiente no qual as máquinas possuem consumos de energia e poder computacional bem variados.

Como pode ser observado na Tabela 3, as máquinas possuem poder computacional variando de 4,7125 *Gflops* até 20,311 *Gflops* e consumo de energia variando de 3250000 a 27410000 *flops/watt*. Já os valores relativos ao consumo de energia das máquinas no estado ocioso, foram definidos com base em valores do *site Energy Star* [ENE12] dependendo do tipo de máquina descrita, variando de 34,57 a 102,1 *watts* por hora (Apêndice A).

Tabela 3 – Configuração das Máquinas.

Máquinas Base	**Máquinas Simulação**				
Nome	Poder (*flops*)	Poder (*Gflops*)	Consumo (*flops/watt*)	*Watts*	Cons. Ocioso (*watts/hora*)
Dual 4 Ghz Intel Zeon	4712500000	4,71	3250000	1450	102,10
2.66 Ghz Pentium IV	7656000000	7,66	3480000	2200	35,53
1.8 Ghz Athlon64	11230000000	11,23	21560000	520,9	34,96
2.0 Ghz Athlon64	12350000000	12,35	22030000	560,9	53,89
2.2 Ghz Athlon64	13400000000	13,40	22020000	608,5	55,53
2.4 Ghz Athlon64	14310000000	14,31	21410000	668,5	56,56
3.6 Ghz Athlon64	19550000000	19,55	27410000	713,2	20,38
4 AMD Opteron 2.2 Ghz	19964000000	19,96	7130000	2800	34,57
2 Quad Core Xeon 2.5 Ghz	20311000000	20,31	10690000	1900	49,00

Extraído de [GAL12], [ENE12] e [SHA06].

Configuração de Consumo dos *Links* de Comunicação

No ambiente de simulação construído, as máquinas comunicam-se através de *links*. São utilizados 191 *links* para permitir o envio de tarefas entre a máquina mestre e as escravas. A fim de simular a distância entre a máquina mestre e uma escrava, a quantidade de *links* varia. Dessa maneira, para conectar uma determinada máquina escrava ao mestre podem ser utilizados 4 *links*, enquanto para conectar outra máquina escrava, podem ser utilizados 10 *links*. Por essa razão, para que fosse possível calcular o consumo no envio de tarefas entre a máquina mestre e a escrava, foi necessário mapear cada um dos *links* utilizados por cada canal de comunicação (Apêndice B). Além disso, também foi preciso, para cada um dos *links* de comunicação utilizados no ambiente de grade simulado, definir valores de consumo de energia. Tais valores são baseados em [MAH09], variando de 2,1 a 13,7 *miliwatts/Mbps* (consumo em *miliwatts* por *megabits* por segundo transferido) e podem ser observados detalhadamente no Apêndice B.

5.3 Análise dos Resultados

Nesta seção, são apresentados os resultados obtidos nos diferentes cenários de testes descritos anteriormente (Apêndice C). O objetivo é realizar uma análise a respeito do comportamento dos algoritmos de escalonamento testados nos quesitos consumo de energia e desempenho.

Plataforma 1

A primeira análise é sobre a plataforma menos heterogênea, a plataforma 1. Para 29 tarefas, como pode ser observado na Figura 16, as versões estáticas do algoritmo LECSA mostraram-se mais eficientes quanto ao consumo de energia do que os demais algoritmos. Com 0% de variação no tamanho das tarefas (heterogeneidade das tarefas), o LECSA, se comparado ao segundo menor algoritmo consumidor de energia (DLECSA) tem um ganho de 314,27%. Quanto ao desempenho, o LECSA tem uma perda de apenas

7,79% em relação aos algoritmos com melhor desempenho, o Suff e o DFPLTF. Para 25, 50 e 75% de variação, o LECSA, LECSA2 e LECSA3 também são melhores em eficiência energética, sempre apresentando perdas pouco significativas em desempenho. Entretanto, para 100% de variação, a melhora em relação ao consumo diminui, pois são usadas 29 máquinas (1 máquina por tarefa), enquanto os demais algoritmos usam uma quantidade menor de máquinas. Este fator afeta na redução do consumo das versões do LECSA, mas aumenta o seu ganho de desempenho. Nesse caso, o desempenho do LECSA3 é um dos melhores.

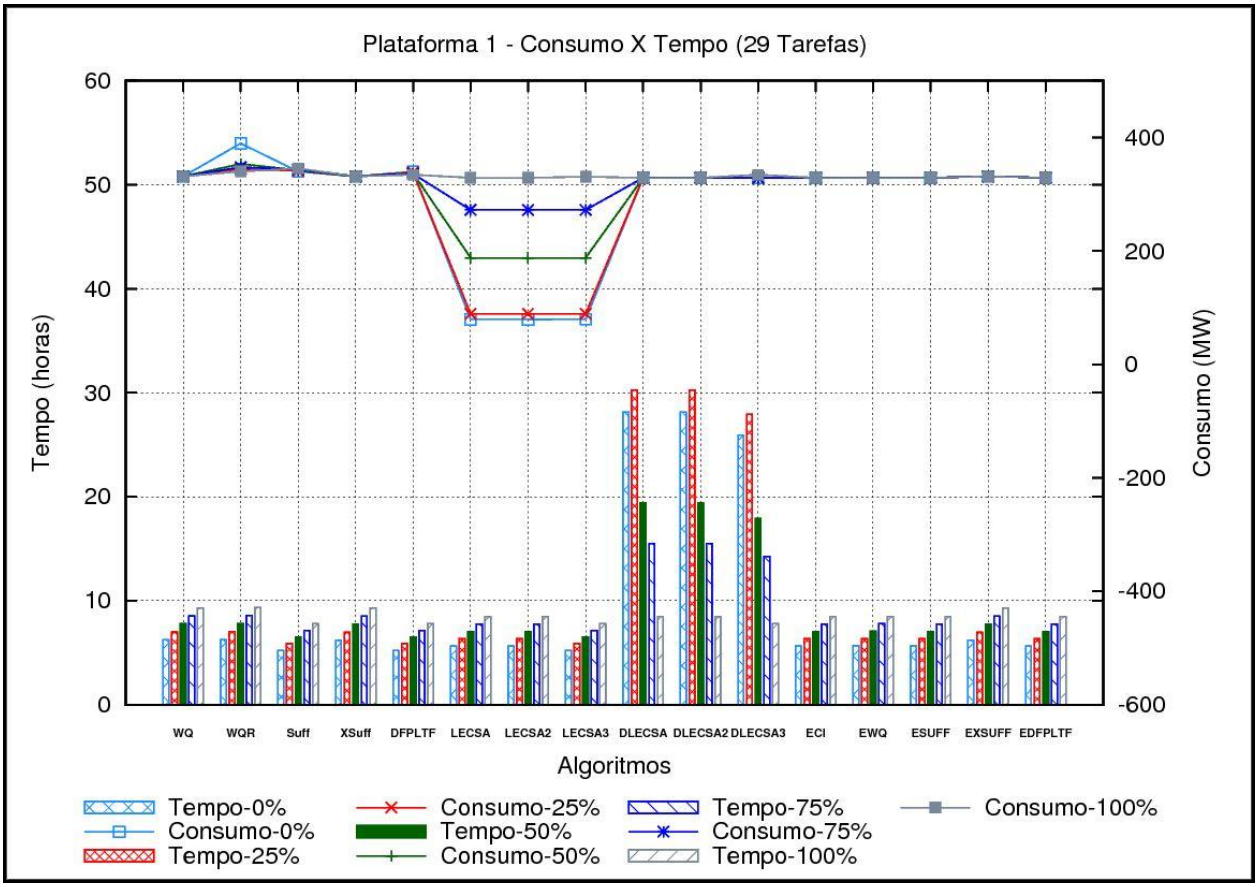

Figura 16 – Plataforma 1 com 29 tarefas.

Como podem ser observados, na Figura 17, para 144 tarefas, os ganhos em relação ao consumo de energia são menores que para 29 tarefas. Para 25% de heterogeneidade das tarefas, a variação com pior perda de desempenho observada, o DLECSA e o DLECSA2 apresentaram uma perda de 76,95% em relação ao melhor algoritmo, o DFPLTF. Entretanto, para 100% de variação, o LECSA registrou uma perda de desempenho bem menor, de apenas 26,47%, com um ganho de 0,9% em redução de consumo de energia, também em comparação com o DFPLTF.

A Figura 18 por sua vez, ilustra o caso com o maior número de tarefas (720) no qual a granularidade da aplicação é baixa, ou seja, há muitas tarefas para poucas máquinas. Tanto as versões estáticas quanto as dinâmicas do LECSA demostraram uma redução no

consumo de energia, que foi de no máximo 18,65% para 25% de variação em comparação com o Suff que foi o pior em eficiência energética. Ao analisar a questão desempenho, a pior perda foi demostrada para 25% de variação pelos algoritmos DLECSA e DLECSA2. A perda foi de 78,77%, em comparação com o algoritmo que obteve o melhor desempenho (EWQ). Já a menor perda de desempenho (19,05%), foi a obtida pelo LECSA2 com 100% de variação com uma economia de energia de 0,46% em comparação com o WQ que para este caso foi o melhor em desempenho.

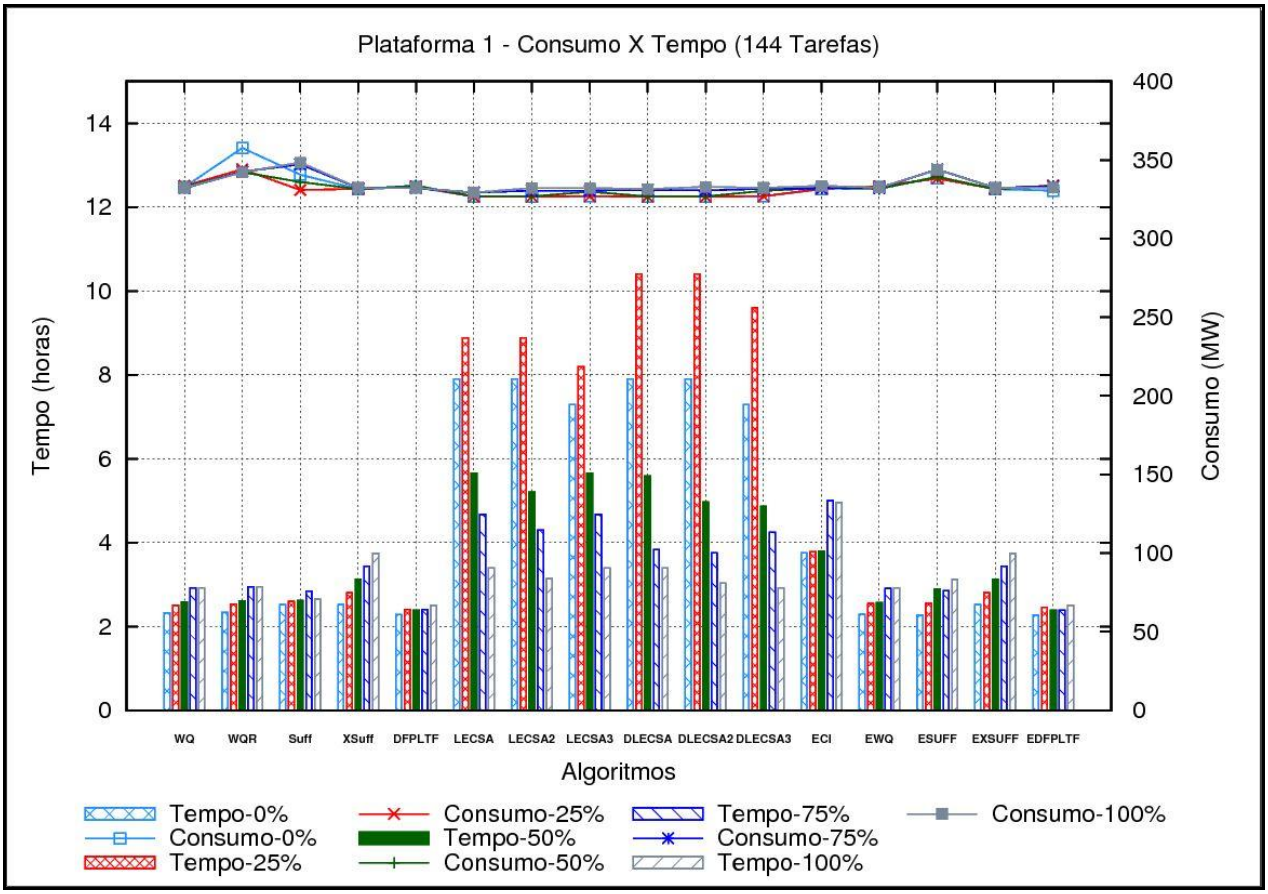

Figura 17 – Plataforma 1 com 144 tarefas.

Plataforma 2

Agora analisaremos a plataforma 2, que possui um nível maior de heterogeneidade. Em virtude disso, os algoritmos desenvolvidos neste trabalho devem apresentar resultados mais expressivos, no quesito eficiência energética, do que os da plataforma 1.

Inicialmente, para 29 tarefas, as três versões estáticas do LECSA obtiveram valores muito representativos quanto à eficiência energética sem grandes perdas em relação ao desempenho como apresenta a Figura 19. Para o caso intermediário, com 50% de heterogeneidade das tarefas, tanto o LECSA quanto o LECSA2, apresentaram uma redução no consumo de energia que atingiu 332,24% em comparação com o pior consumo que foi o obtido pelo algoritmo WQR. Já em relação ao desempenho, para a mesma variação, o LECSA3 foi o melhor, enquanto o LECSA e o LECSA2 apresentaram uma perda de desempenho de apenas 13,61%.

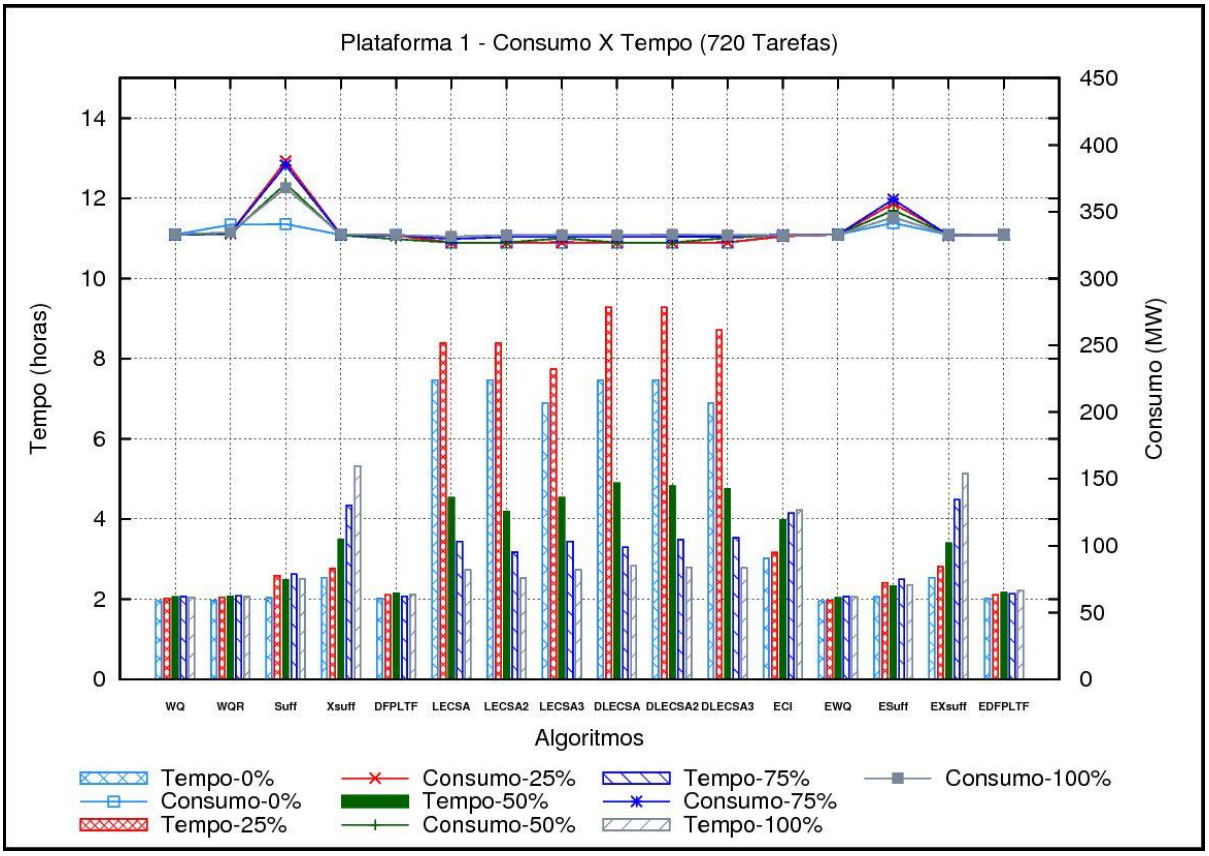

Figura 18 – Plataforma 1 com 720 tarefas.

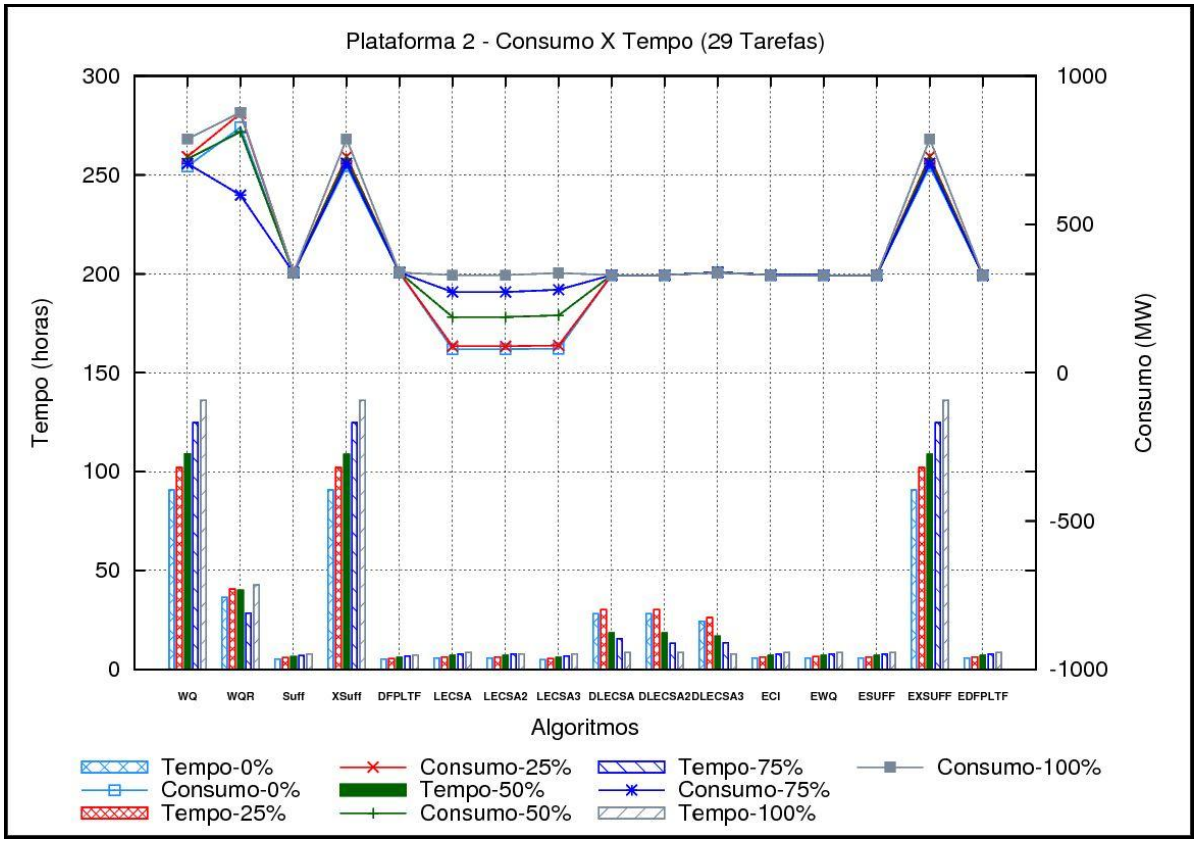

Figura 19 – Plataforma 2 com 29 tarefas.

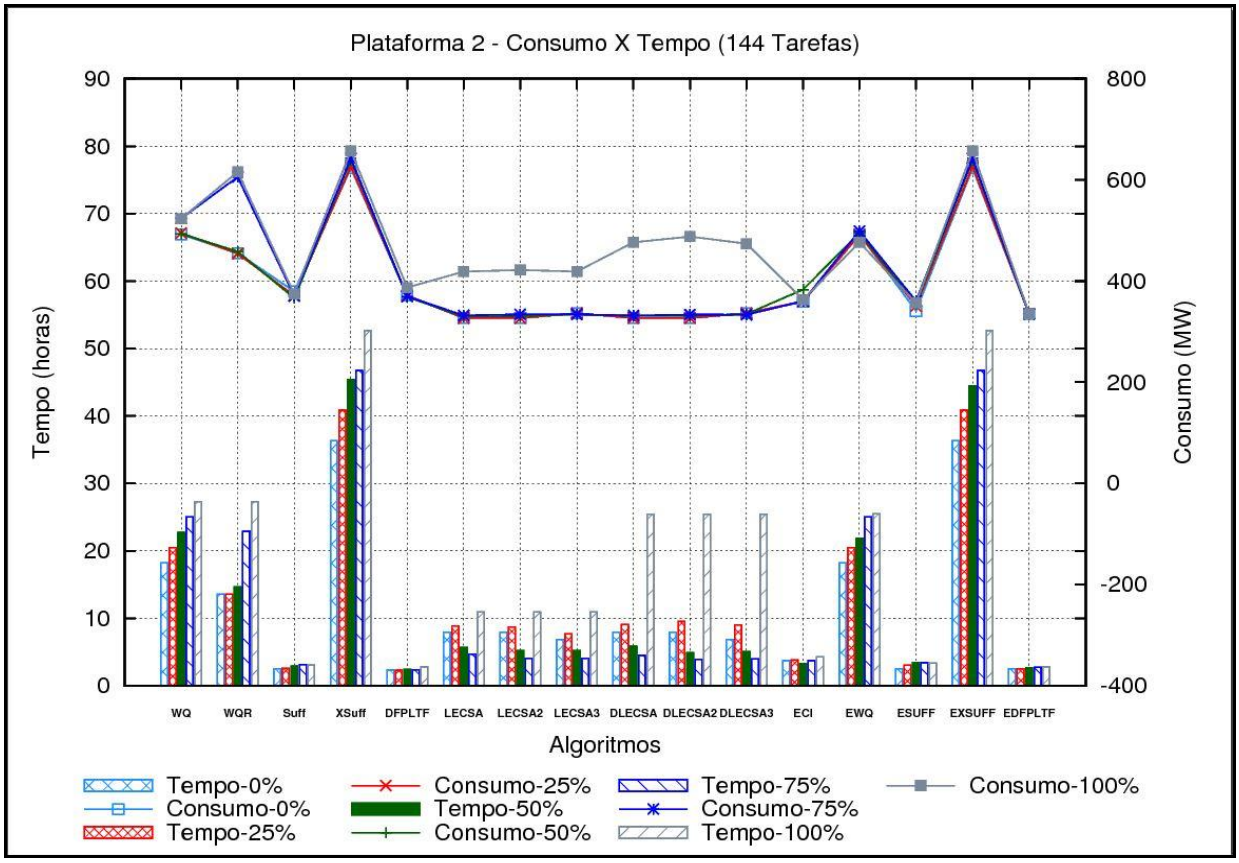

Figura 20 – Plataforma 2 com 144 tarefas.

Com 144 tarefas, como se apresenta na Figura 20, a eficiência energética diminui. Para 100% de variação, é constatada a primeira perda em consumo de energia de todas as versões do LECSA, para alguns dos outros algoritmos testados. Em comparação com o EDFPLTF, que foi o melhor em relação ao consumo, o LECSA apresentou uma perda de 20,52% em eficiência energética e 74,61% em desempenho. Entretanto, para 50% de variação o LECSA apresentou um ganho de 11,77% em relação ao consumo de energia, com perda de 57,70% em desempenho, em comparação com o DFPLTF que apresenta o melhor desempenho. As versões dinâmicas do LECSA também apresentaram uma redução no consumo de energia entre 0 e 75% de heterogeneidade das tarefas.

Para 720 tarefas, como pode ser observado na Figura 21, a perda de eficiência energética permanece para o caso de 100% de heterogeneidade. Entretanto, para 50% o ganho de eficiência energética do LECSA aumentou, em relação ao caso de 144 tarefas, ficando em 31,28%. E a perda de desempenho diminuiu, atingindo os 39,29%, em comparação com o Suff que apresenta o melhor desempenho. Sendo que a perda de desempenho mínima obtida para 720 tarefas foi com 75% de heterogeneidade. Onde a perda ficou em 9,33%, com ganho de 18,56% em relação ao consumo de energia em comparação com o algoritmo DFPLTF que é o melhor em termos de desempenho para o caso citado.

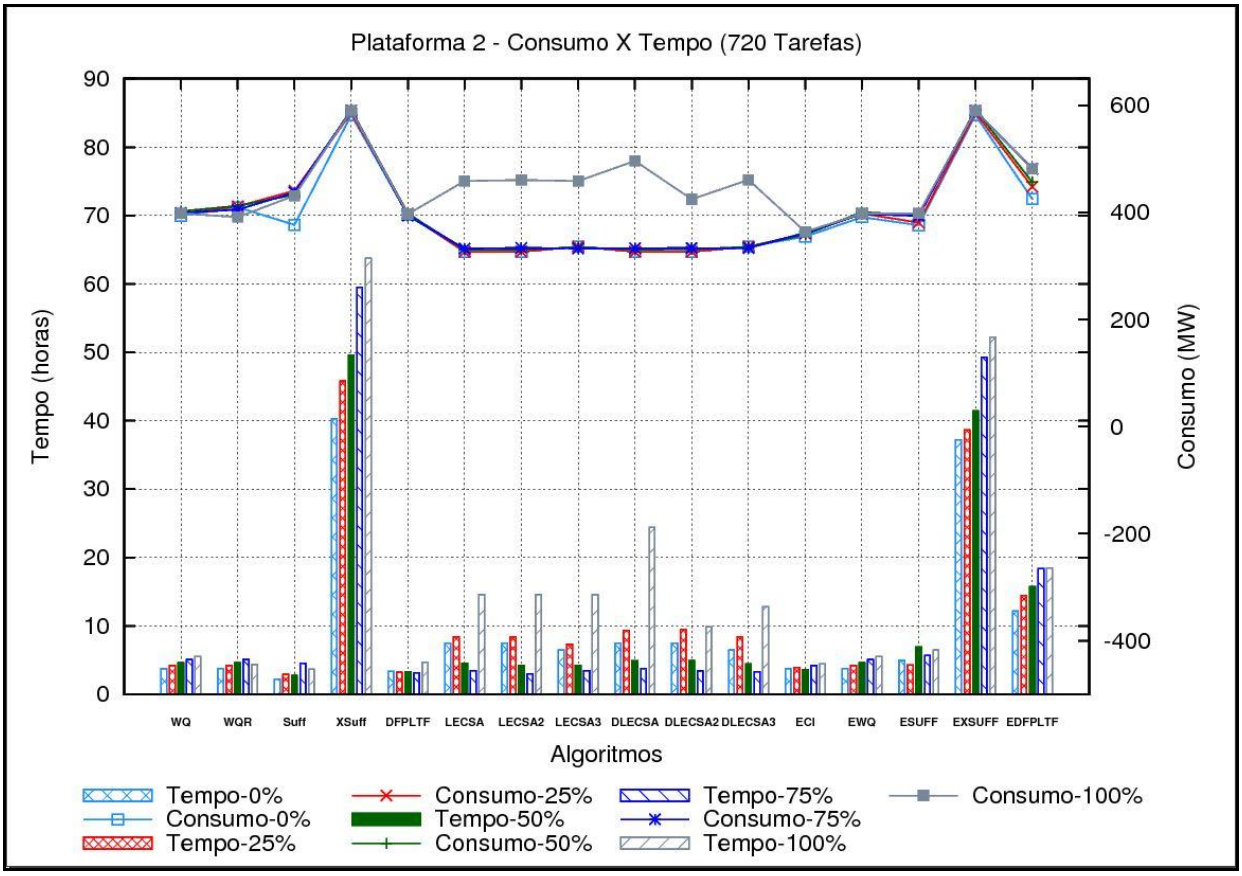

Figura 21 – Plataforma 2 com 720 tarefas.

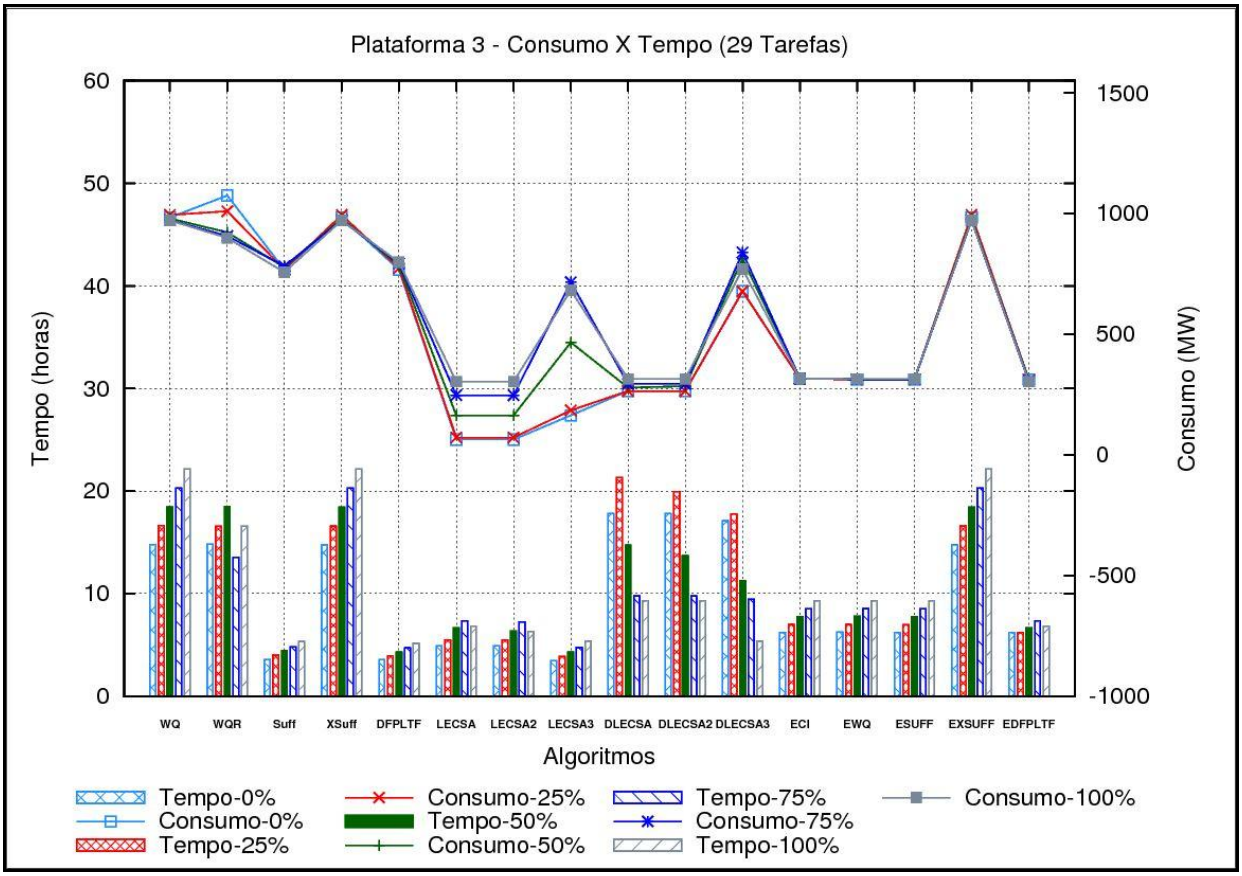

Figura 22 – Plataforma 3 com 29 tarefas.

Plataforma 3

O último caso a ser analisado corresponde ao da plataforma 3, que possui o maior nível de heterogeneidade de máquinas, e portanto, deve apresentar os maiores ganhos de eficiência energética em comparação com os alcançados na plataforma 1 e 2. Esta grande heterogeneidade de máquinas também faz com que este caso possa ser melhor explorado pelos algoritmos desenvolvidos neste trabalho.

Para 29 tarefas, como mostra a Figura 22, há ganho em eficiência energética para todas as variações na heterogeneidade das tarefas. Por exemplo, com 25% de heterogeneidade o LECSA e o LECSA2 apresentam um ganho de 156,40% em eficiência energética, com apenas 29,51% de perda em desempenho, em comparação com o LECSA3, que é o melhor algoritmo em termos de desempenho. Já com 100%, em comparação com o DFPLTF, o melhor algoritmo em termos de desempenho, há um ganho de 162,92% em eficiência energética e uma perda de apenas 18,51% em desempenho.

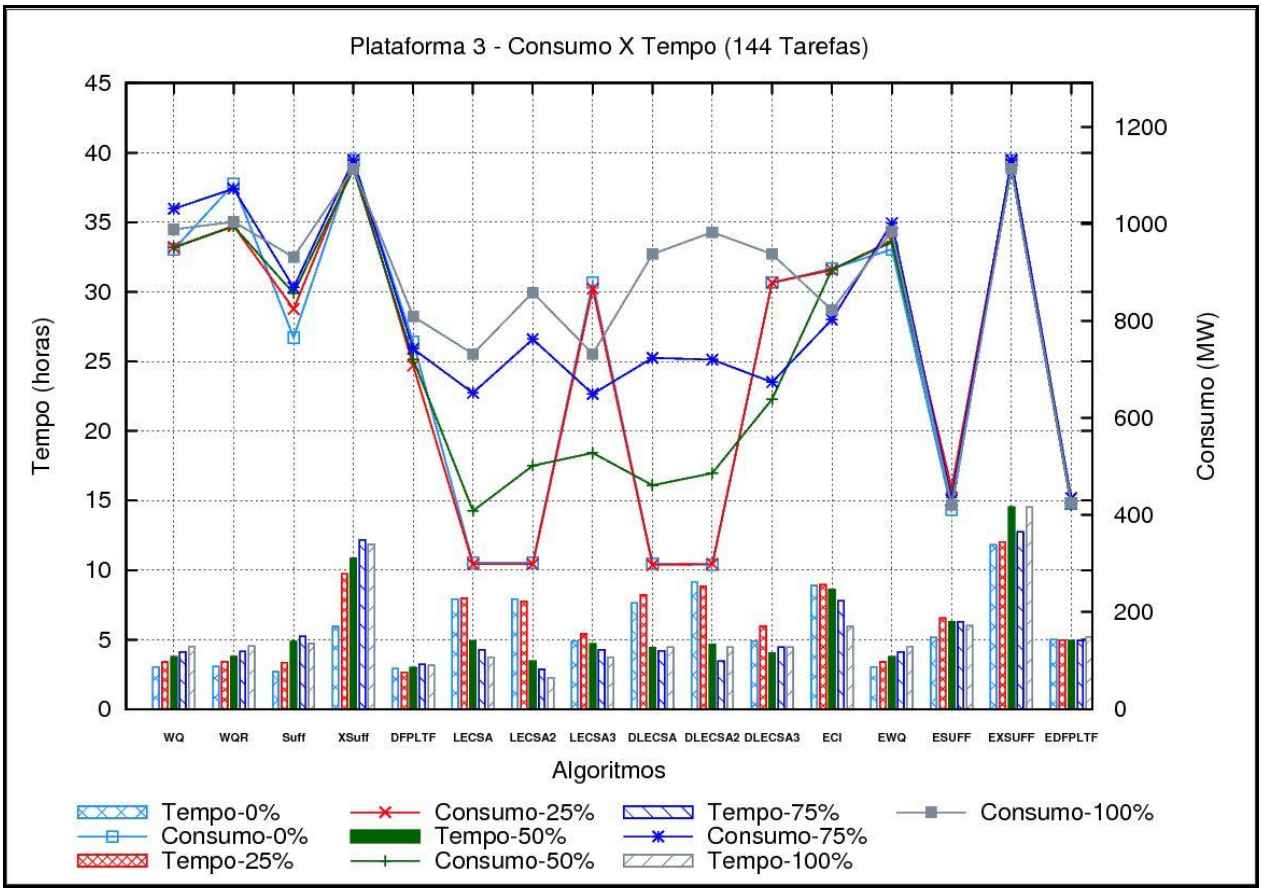

Figura 23 – Plataforma 3 com 144 tarefas.

No caso de 144 tarefas, como mostra a Figura 23, para o caso intermediário, com 50% de heterogeneidade, o ganho de eficiência energética do LECSA atinge 76,46%. Com uma perda de desempenho de 38,93%, em comparação com o algoritmo DFPLTF, que é o melhor em desempenho. Entretanto, para 100% de variação, o algoritmo ESUFF é o que apresenta o menor consumo de energia, consumindo 92,27% a menos que o

DFPLTF (o melhor em desempenho). Com uma perda de apenas 47,41% em desempenho. Porém, se o objetivo fosse obter uma perda menor em desempenho e ganhar menos em eficiência energética, o LECSA e o LECSA3 seriam boas alternativas. Pois, para esse mesmo caso, a perda de desempenho seria de 15,77% e o ganho em consumo de energia de 10,56%.

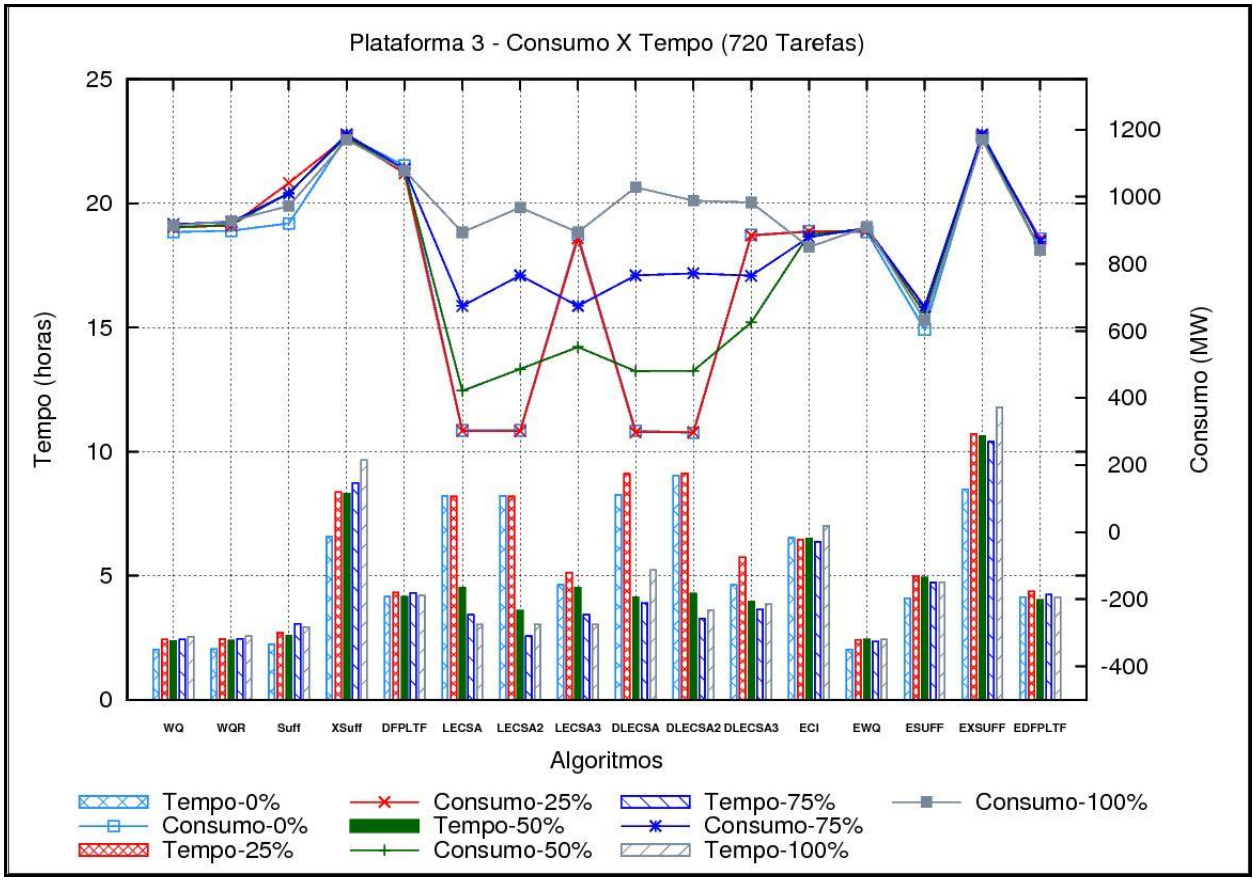

Figura 24 – Plataforma 3 com 720 tarefas.

O último caso a ser analisado é o caso de 720 tarefas, Figura 24. Para 50% de heterogeneidade, o LECSA apresentou uma redução de 115,63% no consumo de energia, com 47,68% de perda em desempenho em comparação com o WQ que é o melhor em termos de desempenho. Com essa heterogeneidade, todos os algoritmos desenvolvidos neste trabalho (LECSA, LECSA2, LECSA3, DLECSA, DLECSA2 e DLECSA3) apresentaram uma eficiência energética acima de 45%. Já para 100% de variação, o melhor em termos de eficiência energética foi o ESUFF com 43,99% de redução de consumo e 48,63% de perda de desempenho em comparação com o EWQ que é o algoritmo que obteve o melhor desempenho. No entanto, se a intenção for não prejudicar tanto o desempenho, uma opção seria utilizar ou o algoritmo LECSA, ou o algoritmo LECSA3. Ambos os algoritmos para o mesmo caso apresentado acima possuem um ganho de 29,32% em redução de consumo de energia e perda de apenas 19,81% em desempenho.

Tabela 4 – Melhor Algoritmo em Desempenho x Melhor Algoritmo em Eficiência Energética.

		Consumo				Desempenho			
	Tarefas	Heterog.	Melhor	Alg. Comparado	Dif. %	Heterog.	Melhor	Alg. Comparado	Dif. %
Plataforma 1	29	0%	LECSA2	DFPLTF	+ 328,24	0%	DFPLTF	LECSA2	- 7,81
		25%	LECSA2	DFPLTF	+ 279,13	25%	DFPLTF	LECSA2	- 7,81
		50%	LECSA	LECSA3	+ 0,05	50%	LECSA3	LECSA	- 7,80
		75%	LECSA2	DFPLTF	+ 26,19	75%	DFPLTF	LECSA2	- 7,81
		100%	LECSA	DLECSA	+ 1,65	100%	DLECSA	LECSA	- 7,81
	144	0%	LECSA2	ESUFF	+ 3,55	0%	ESUFF	LECSA2	- 71,21
		25%	LECSA2	DFPLTF	+ 1,95	25%	DFPLTF	LECSA2	- 72,96
		50%	LECSA	DFPLTF	+ 2,12	50%	ESUFF	LECSA	- 57,71
		75%	LECSA	EDFPLTF	+ 1,17	75%	EDFPLTF	LECSA	- 48,81
		100%	LECSA	DFPLTF	+ 0,92	100%	DFPLTF	LECSA	- 26,47
	720	0%	LECSA2	WQ	+ 1,86	0%	WQ	LECSA2	- 73,94
		25%	LECSA2	EWQ	+ 1,85	25%	EWQ	LECSA2	- 76,47
		50%	LECSA	EWQ	+ 1,84	50%	EWQ	LECSA	- 55,27
		75%	LECSA	WQ	+ 0,97	75%	WQ	LECSA	- 40,10
		100%	LECSA	WQ	+ 0,46	100%	WQ	LECSA	- 25,37
Plataforma 2	29	0%	LECSA2	DFPLTF	+ 325,09	0%	DFPLTF	LECSA2	- 7,68
		25%	LECSA2	LECSA3	+ 2,90	25%	LECSA3	LECSA2	- 13,64
		50%	LECSA2	LECSA3	+ 2,90	50%	LECSA3	LECSA2	- 13,64
		75%	LECSA2	LECSA3	+ 2,89	75%	LECSA3	LECSA2	- 13,65
		100%	LECSA	DFPLTF	+ 2,74	100%	DFPLTF	LECSA	- 13,66
	144	0%	LECSA2	DFPLTF	+ 13,65	0%	DFPLTF	LECSA2	- 71,07
		25%	LECSA2	DFPLTF	+ 13,63	25%	DFPLTF	LECSA2	- 73,76
		50%	LECSA	DFPLTF	+ 11,77	50%	DFPLTF	LECSA	- 57,70
		75%	LECSA	DFPLTF	+ 11,20	75%	DFPLTF	LECSA	- 48,89
		100%	EDFPLTF	DFPLTF	+ 15,55	100%	DFPLTF	EDFPLTF	- 0,29
	720	0%	LECSA2	Suff	+ 15,36	0%	Suff	LECSA2	- 70,98
		25%	LECSA2	Suff	+ 34,84	25%	Suff	LECSA2	- 65,61
		50%	LECSA	Suff	+ 31,28	50%	Suff	LECSA	- 39,29
		75%	LECSA	DFPLTF	+ 18,56	75%	DFPLTF	LECSA	- 9,33
		100%	ECI	Suff	+ 18,56	100%	Suff	ECI	-17,57
Plataforma 3	29	0%	LECSA2	Suff	+ 1102,56	0%	Suff	LECSA2	- 26,56
		25%	LECSA2	LECSA3	+ 156,41	25%	LECSA3	LECSA2	- 29,51
		50%	LECSA2	DFPLTF	+ 372,80	50%	DFPLTF	LECSA2	- 32,16
		75%	LECSA2	DFPLTF	+ 221,03	75%	DFPLTF	LECSA2	- 34,60
		100%	LECSA2	DFPLTF	+ 219,68	100%	DFPLTF	LECSA2	- 18,53
	144	0%	DLECSA2	ESUFF	+ 157,74	0%	ESUFF	DLECSA2	- 70,53
		25%	DLECSA	DFPLTF	+ 138,31	25%	DFPLTF	DLECSA	- 67,57
		50%	DLECSA	DFPLTF	+ 76,46	50%	DFPLTF	DLECSA	- 38,93
		75%	ESUFF	DFPLTF	+ 72,28	75%	DFPLTF	ESUFF	- 48,64
		100%	ESUFF	DFPLTF	+ 92,27	100%	DFPLTF	ESUFF	- 47,41
	720	0%	DLECSA2	EWQ	+ 202,25	0%	EWQ	DLECSA2	- 77,61
		25%	DLECSA	EWQ	+ 202,15	25%	EWQ	DLECSA	- 73,56
		50%	LECSA	WQ	+ 115,63	50%	WQ	LECSA	- 47,68
		75%	ESUFF	EWQ	+ 34,98	75%	EWQ	ESUFF	- 50,44
		100%	ESUFF	EWQ	+ 44,00	100%	EWQ	ESUFF	- 48,63

Depois de terem sido apresentados os resultados gráfico por gráfico, agora o objetivo é apresentar uma avaliação global dos resultados obtidos para que seja possível analisar qual o comportamento dos algoritmos testados em relação à granularidade das aplicações, a heterogeneidade das máquinas e heterogeneidade das tarefas.

A seguir, apresentam-se dados extraídos da análise global dos resultados obtidos. Os dados ilustrados pela Tabela 4 realizam uma comparação do melhor algoritmo em termos de desempenho com o melhor algoritmo em termos de consumo de energia para cada cenário:

- O caso que obteve a maior economia de energia (1102,56%) foi na plataforma 3 com 29 tarefas e 0% de heterogeneidade de tarefas, com uma perda de 26,56% em desempenho (valores obtidos pelo algoritmo LECSA2 em comparação com o melhor algoritmo em termos de desempenho (Suff);
- O caso em que houve menor economia de energia (0,05%) foi na plataforma 1 com 29 tarefas e 50% de heterogeneidade de tarefas, com uma perda de 7,08% em desempenho (valores obtidos pelo LECSA em comparação com o melhor algoritmo em termos de desempenho, que foi a versão 3 do próprio LECSA);
- O caso que atingiu a menor perda de desempenho (0,29%) foi na plataforma 2, com 144 tarefas e 100% de heterogeneidade de tarefas, com redução de 15,55% no consumo de energia (valores obtidos pelo algoritmo EDFPLTF em comparação com sua versão original);
- O caso em que houve a maior perda de desempenho foi do DLECSA2 na plataforma 3 para 720 tarefas com 0% de heterogeneidade de tarefas, atingindo uma perda de 77,61%, em comparação com o EWQ (o melhor em desempenho). Porém, o ganho em eficiência energética atingiu 202,25%;
- Com 29 tarefas o LECSA2 apresentou para a plataforma 1, 2 e 3, os melhores resultados em termos de maior redução no consumo de energia;
- Quanto maior for à heterogeneidade dos *hosts*, maior é a redução no consumo de energia, entretanto, há maior perda de desempenho;
- Quanto maior for à granularidade da aplicação (29 tarefas de 250 *Tflops*), maior é a economia de energia (no máximo 1102,56%), entretanto, há uma perda de desempenho, que é no máximo de 34,60%;
- Em relação à heterogeneidade das tarefas, para a plataforma 1 e 3, quanto mais homogêneas (0%) forem as tarefas, maior é a redução no consumo de energia. Quando as tarefas são 100% heterogêneas a redução de consumo de energia fica entre 0,46% e 219,68%;

- O algoritmo que obteve o melhor desempenho em um número maior de casos, foi o DFPLTF, que aparece em 22 dos 45 casos totais. O segundo colocado é o EWQ, que aparece em 6 dos 45 casos e em terceiro lugar, aparecem empatados o LECSA3 e o Suff, com 5 ocorrências em 45 casos;
- O algoritmo que obteve a melhor eficiência energética em um número maior de casos foi o LECSA2, que apareceu em 19 dos 45 casos totais. O segundo colocado é o LECSA, que aparece em 14 dos 45 casos e em terceiro lugar aparece o ESUFF, com 4 ocorrências em 45 casos.

Quanto aos algoritmos WQ, Suff, XSuff e DFPLTF, que foram modificados para que melhorassem a sua eficiência energética (Seções 4.3.7, 4.3.8, 4.3.9 e 4.3.10), as versões que globalmente obtiveram as maiores reduções foram: ESUFF, EWQ e EDFPLTF. Sendo que, na plataforma 3, as reduções no consumo de energia foram bem mais significativas. Na grande maioria dos casos, o consumo de energia foi reduzido pela metade, com ganho também em termos de desempenho.

De modo geral, os algoritmos desenvolvidos neste trabalho, em especial o LECSA, o LECSA2 e o DLECSA, obtiveram ótimos resultados quanto à eficiência energética, com perdas aceitáveis em desempenho. Como dito no decorrer deste trabalho, não existe um algoritmo perfeito, que apresente o melhor resultado para qualquer tipo de cenário. Dependendo das características das máquinas, das tarefas, da rede de comunicação, determinado algoritmo se sobressairá aos demais. Em virtude disso, dá-se a importância em se conhecer as características do ambiente de grade ao qual o algoritmo de escalonamento será voltado.

6 CONCLUSÃO E TRABALHOS FUTUROS

Esta Dissertação teve como objetivos principais: (i) desenvolvimento do módulo de cálculo de consumo de energia, apresentado na Seção 4.2; (ii) desenvolvimento de algoritmos de escalonamento energeticamente eficientes, apresentados na Seção 4.3. Inicialmente, o Capítulo 2 descreve algumas técnicas de escalonamento de tarefas existentes, em seguida apresenta-se o ambiente de simulação utilizado, o SimGrid, juntamente com a arquitetura da LIBTS. Tal arquitetura foi modificada para que pudessem ser atingidos os objetivos deste trabalho, descritos no Capítulo 4.

Como ponto de partida para os desenvolvimentos descritos nesta Dissertação, realizou-se um estudo referente às atuais abordagens propostas para melhorar a eficiência energética em ambientes de grade computacional. Após esse estudo, iniciaram-se as modificações na arquitetura original da LIBTS, para adicionar a informação de consumo de energia de cada algoritmo na execução das aplicações. Sendo esta a primeira contribuição desta Dissertação.

A segunda contribuição deste trabalho deu-se com o desenvolvimento do módulo de cálculo de consumo de energia, que trouxe algumas ideias sobre técnicas de escalonamento de tarefas que poderiam potencialmente reduzir o consumo de energia. A partir disso, desenvolveu-se o LECSA, totalmente direcionado para eficiência energética, fato que em alguns casos ocasionava uma perda muito significativa em termos de desempenho. Por essa razão, na Seção 4.3, foram descritas as outras duas versões do LECSA, todas estáticas. O desenvolvimento dos três algoritmos de escalonamento estáticos resultou na terceira contribuição deste trabalho. Como os demais algoritmos, WQ, Suff, XSuff, DFPLTF, ECI, são dinâmicos, a quarta contribuição foi a implementação das versões dinâmicas do LECSA, para fim de comparações de consumo de energia e desempenho, apresentadas na Seção 5.3.

A contribuição mais relevante deste trabalho é a validação do módulo de consumo e dos algoritmos de escalonamento energeticamente eficientes, para uso na ferramenta SimGrid. O LECSA, o LECSA2 e o DLECSA obtiveram resultados muito significativos em eficiência energética, com perdas aceitáveis em termos de desempenho, principalmente quando o ambiente computacional apresentar alto nível de heterogeneidade entre as máquinas da grade. Com a análise dos resultados obtidos, pode-se identificar qual algoritmo de escalonamento se torna mais adequado para ganho em eficiência energética ou em desempenho. Os resultados apresentados nesta Dissertação impulsionaram a proposta de desenvolvimento dos trabalhos futuros.

Uma versão inicial da implementação do módulo de cálculo de consumo e do algoritmo LECSA, rendeu uma aprovação para publicação no ACM-SAC 2013, intitulada:

"*Energy Efficiency Management in Computational Grids through Energy-aware Scheduling*" [TEO13].

6.1 Trabalhos Futuros

Como trabalho futuro propõe-se o desenvolvimento de um escalonador global de aplicações. Cujo objetivo é que o escalonador seja capaz de decidir qual é a melhor opção de escalonamento em termos ou de redução de consumo de energia ou de desempenho, ou uma boa relação entre consumo e desempenho, conforme a escolha feita pelo usuário. Para que o escalonador seja capaz de tomar decisões de escalonamento ele deverá se basear nas características das tarefas e do ambiente computacional. Para tal, assim que uma nova aplicação chegar para ser escalonada será necessário verificar a característica da mesma, e em conjunto com as informações das máquinas que compõem a grade, decidir qual é o algoritmo de escalonamento mais adequado para aquele caso. Este analisador do perfil de aplicações deverá ser baseado em Redes Neurais Artificiais ou Algoritmos Genéticos, pois para decidir qual é o melhor algoritmo de escalonamento para determinada necessidade (desempenho, consumo, ou ambos) deve-se passar pela análise de um histórico de execuções.

O desenvolvimento do escalonador citado acima abrange as seguintes atividades:

- Melhoria do módulo de cálculo de consumo de energia;
- Implementação de novos algoritmos de escalonamento de tarefas para grades computacionais voltados para desempenho e economia de energia;
- Criação de novos cenários de testes;
- Realização de novos experimentos para validar os desenvolvimentos realizados.

REFERÊNCIAS

[BUY02] BUYYA, R.; KRAUTER, K.; MAHESWARAN, M. “A Taxonomy and Survey of Grid Resource Management Systems for Distributed Computing”. Software-Practice and Experience, vol.32, Fev 2002, pp. 135–164.

[CAS00] CASANOVA, H.; ZAGORODNOV, D.; BERMAN, F.; LEGRAND, A. “Heuristics for Scheduling Parameter Sweep Applications in Grid Environments”. In: Proceedings of the 9th Heterogeneous Computing Workshop (HCW ’00), IEEE Computer Society, Washington, USA, 2000, 15p.

[CAS88] CASAVANT, T.; KUHL, J. “A Taxonomy of Scheduling in General-purpose Distributed Computing Systems”. IEEE Transactions on Software Engineering, vol. 14, Fev 1988, pp. 141-154.

[ENE12] Energy Star. “Find Energy Star Products”. Capturado em: http://www.energystar.gov/index.cfm?fuseaction=find_a_product.showProductGroup&pgw_code=CO, Outubro 2012.

[FEN03] FENG, W. “Making a Case for Efficient Supercomputing”. ACMQueue, vol. 1, Out 2003, pp. 54-64.

[FEN07] FENG, W.; CAMERON, K. “The Green500 List: Encouraging Sustainable Supercomputing”. Computer, vol. 40, Dez 2007, pp. 50-55.

[FRA11] FRANCO, P. B. “Escalonamento de Tarefas em Ambiente de Simulação de Grid Computacional”. Dissertação de Mestrado em Ciência da Computação, Universidade Estadual paulista “Júlio de Mesquita Filho”, UNESP, 2011, 100p.

[GAL12] GALIZIA, A.; QUARATI, A. “Job Allocation Strategies for Energy-aware and Efficient Grid Infrastructures”. Journal of Systems and Software, vol. 75, Jul 2012, pp. 1588-1606.

[GAR09] GARG, S. K.; BUYYA R. “Exploiting Heterogeneity in Grid Computing for Energy-Efficient Resource Allocation”. In: 17th International Conference on Advanced Computing and Communications (ADCOM), 2009, Bengaluru, India, 7p.

[KIM07] KIM, K. H.; BUYYA, R.; KIM, J. “Power Aware Scheduling of Bag-of-Tasks Applications with Deadline Constraints on DVS-enabled Clusters”. In: Proceedings of the Seventh IEEE International Symposium on Cluster Computing and the Grid (CCGRID '07), IEEE Computer Society, Washington, USA, 2007, pp. 541-548.

[LAM09] LAMMIE, M.; BRENNER, P.; THAIN, D. “Scheduling Grid Workloads on Multicore Clusters to Minimize Energy and Maximize Performance”. In: 10th IEEE/ACM International Conference on Grid Computing, Out 2009, pp. 145-152.

[LIU10] LIU, Y.; ZHU, H. “A Survey of the Research on Power Management Techniques for High-Performance Systems". Software Practice & Experience, vol. 40, Out 2010, pp. 943-964.

[MAH09] MAHADEVAN, P.; SHARMA, P.; BANERJEE, S.; RANGANATHAN, P. “A Power Benchmarking Framework for Network Devices”. In: Proceedings of the 8th International IFIP-TC 6 Networking Conference (NETWORKING '09), Springer-Verlag, Berlin, Heidelberg, 2009, pp. 795-808.

[MAM11] MÄMMELÄ, O.; MAJANEN, M.; BASMADJIAN, R.; MEER, H. De; GIESLER, A.; HOMBERG, W. “Energy-aware Job Scheduler for High-performance Computing”. Computer Science-Research and Development, Springer Berlin/Heidelberg, vol. 27, Nov 2012, pp. 1865-2034.

[MEN95] MENASCÉ, D. A.; SAHA, D.; PORTO, S. C. da S.; ALMEIDA, V. A. F.; TRIPATHI, S. K. “Static and Dynamic Processor Scheduling Disciplines in Heterogeneous Parallel Architectures”. Journal of Parallel and Distributed Computing, vol. 28, Jul 1995, pp. 1-18.

[MON12] MONTES, A. F.; ABRIL, L. G.; ORTEGA, J. A.; LEFÈVRE, L. “Smart Scheduling for Saving Energy in Grid Computing”. Expert Systems with Applications, vol. 39, Ago 2012, pp. 9443-9450.

[MUR02] MURSHED, M.; BUYYA, R. “Using the GridSim Toolkit for Enabling Grid Computing Education”. In: International Conference on Communication Networks and Distributed Systems Modeling and Simulation, 2002, 7p.

[NEM11] NEMETZ, R. “Otimizando o Fluxo de Tarefas em Sistemas Distribuídos de Impressão: um Algoritmo de Escalonamento Dinâmico Não Preemptivo Baseado em Mecanismo de Previsão”. Dissertação de Mestrado, Programa de Pós-Graduação em Ciência da Computação, PUCRS, 2011, 79p.

[PON10] PONCIANO, L.; BRASILEIRO, F.; SANTANA, J. V.; CARVALHO, M.; GAUDENCIO, M. "Usando as Estratégias Sobreaviso e Hibernação para Economizar Energia em Grades Computacionais Oportunistas". Revista Brasileira de Redes de Computadores e Sistemas Distribuídos, vol. 3, 2010, pp. 9-20.

[REI05] REIS, V. Q. dos. “Escalonamento em Grids Computacionais: Estudo de Caso”. Monografia, Instituto de Ciências Matemáticas e de Computação, USP, 2005, 94p.

[SCA07] SCARAMELLA, J.; HEALEY, M. “Service-Based Approaches to Improving Data Center Thermal and Power Efficiencies”. Capturado em: https://h30406.www3.hp.com/campaigns/2007/promo/13ISN9/images/_3pageFINAL_TLT_Services_paper_52407.pdf, Maio 2011.

[SHA06] SHARMA, S.; CHUNG-HSING, H.; WU-CHUN, F. "Making a Case for a Green500 List". In: 20th International Parallel and Distributed Processing Symposium, IPDPS, 2006, pp. 25-29.

[SIL03] SILVA, D. P. da; CIRNE, W.; BRASILEIRO, F. V. “Trading Cycles for Information: Using replication to Schedule Bag-of-Tasks Applications on Computational Grids”. In: Euro-Par Parallel Processing, 2003, pp. 169-180.

[SIL09] SILVA, F. A. B. da; SENGER, H. “Improving Scalability of Bag-of-Tasks Applications Running on Master-slave Platforms”. Parallel Computing, vol. 35, Fev 2009, pp. 57-71.

[SIM12] SimGrid. “SimGrid 3.5: Documentation”. Capturado em: http://simgrid.gforge.inria.fr/simgrid/3.5/doc/, Junho 2012.

[TAN08] TANEMBAUM, A.; WOODHULL, A. “Sistemas Operacionais, Projeto e Implementação”. Tradução João Tortello. 3 ed. Porto Alegre: Bookman, 2008, 992p.

[TEO13] TEODORO, S.; CARMO, A. B.; FERNANDES, L. G. “Energy Efficiency Management in Computational Grids through Energy-aware Scheduling”. In: 28th Symposium On Applied Computing (ACM SAC), Coimbra, Portugal, 2006, pp.1163-1168.

[TOP11] Top500. “Top500 Supercomputers Sites”. Capturado em: http://www.top500.org /list/2011/11/100, Novembro 2011.

[WAN11] WANG, L.; LASZEWSKI, G. von; HUANG, F.; DAYAL, J.; FRULANI, T.; FOX, G. “Task Scheduling with ANN-based Temperature Prediction in a Data Center: a Simulation-based Study”. Engineering with Computers, vol. 27, Out 2011, pp. 381-391.

APÊNDICE A – HETEROGENEIDADE E CONSUMO DAS MÁQUINAS

Plataforma 1				
Máquinas	**Poder (*flops*)**	***flops/watt***	***Watts***	**Cons. ocioso/hora**
0	11230000000	21560000	520,9	34,96
1	12350000000	22030000	560,5	53,89
2	13400000000	22020000	608,5	55,53
3	11230000000	21560000	520,9	34,96
4	12350000000	22030000	560,5	53,89
5	13400000000	22020000	608,5	55,53
6	13400000000	22020000	608,5	55,53
7	11230000000	21560000	520,9	34,96
8	12350000000	22030000	560,5	53,89
9	11230000000	21560000	520,9	34,96
10	12350000000	22030000	560,5	53,89
11	13400000000	22020000	608,5	55,53
12	11230000000	21560000	520,9	34,96
13	12350000000	22030000	560,5	53,89
14	13400000000	22020000	608,5	55,53
15	13400000000	22020000	608,5	55,53
16	11230000000	21560000	520,9	34,96
17	12350000000	22030000	560,5	53,89
18	13400000000	22020000	608,5	55,53
19	13400000000	22020000	608,5	55,53
20	11230000000	21560000	520,9	34,96
21	12350000000	22030000	560,5	53,89
22	11230000000	21560000	520,9	34,96
23	12350000000	22030000	560,5	53,89
24	13400000000	22020000	608,5	55,53
25	11230000000	21560000	520,9	34,96
26	11230000000	21560000	520,9	34,96
27	12350000000	22030000	560,5	53,89
28	13400000000	22020000	608,5	55,53
29	11230000000	21560000	520,9	34,96
30	12350000000	22030000	560,5	53,89
31	13400000000	22020000	608,5	55,53
32	13400000000	22020000	608,5	55,53
33	11230000000	21560000	520,9	34,96
34	12350000000	22030000	560,5	53,89
35	11230000000	21560000	520,9	34,96
36	12350000000	22030000	560,5	53,89
37	13400000000	22020000	608,5	55,53
38	11230000000	21560000	520,9	34,96
39	12350000000	22030000	560,5	53,89
40	13400000000	22020000	608,5	55,53
41	13400000000	22020000	608,5	55,53
42	11230000000	21560000	520,9	34,96

43	12350000000	22030000	560,5	53,89
44	13400000000	22020000	608,5	55,53
45	13400000000	22020000	608,5	55,53
46	11230000000	21560000	520,9	34,96
47	12350000000	22030000	560,5	53,89
48	11230000000	21560000	520,9	34,96
49	12350000000	22030000	560,5	53,89
50	13400000000	22020000	608,5	55,53
51	11230000000	21560000	520,9	34,96
52	11230000000	21560000	520,9	34,96
53	12350000000	22030000	560,5	53,89
54	13400000000	22020000	608,5	55,53
55	11230000000	21560000	520,9	34,96
56	12350000000	22030000	560,5	53,89
57	13400000000	22020000	608,5	55,53
58	13400000000	22020000	608,5	55,53
59	11230000000	21560000	520,9	34,96
60	12350000000	22030000	560,5	53,89
61	11230000000	21560000	520,9	34,96
62	12350000000	22030000	560,5	53,89
63	13400000000	22020000	608,5	55,53
64	11230000000	21560000	520,9	34,96
65	12350000000	22030000	560,5	53,89
66	13400000000	22020000	608,5	55,53
67	13400000000	22020000	608,5	55,53
68	11230000000	21560000	520,9	34,96
69	12350000000	22030000	560,5	53,89
70	13400000000	22020000	608,5	55,53
71	13400000000	22020000	608,5	55,53
72	11230000000	21560000	520,9	34,96
73	12350000000	22030000	560,5	53,89
74	11230000000	21560000	520,9	34,96
75	12350000000	22030000	560,5	53,89
76	13400000000	22020000	608,5	55,53
77	11230000000	21560000	520,9	34,96
78	11230000000	21560000	520,9	34,96
79	12350000000	22030000	560,5	53,89
80	13400000000	22020000	608,5	55,53
81	13400000000	22020000	608,5	55,53
82	11230000000	21560000	520,9	34,96
83	12350000000	22030000	560,5	53,89
84	13400000000	22020000	608,5	55,53
85	13400000000	22020000	608,5	55,53
86	11230000000	21560000	520,9	34,96
87	12350000000	22030000	560,5	53,89
88	13400000000	22020000	608,5	55,53
89	13400000000	22020000	608,5	55,53
Média	**12,36 Gflops**			

Plataforma 2				
Máquinas	**Poder (*flops*)**	***flops/watt***	***Watts***	**Cons. ocioso/hora**
0	7656000000	3480000	2200	35,53
1	11230000000	21560000	520,9	34,96
2	14310000000	21410000	668,5	56,56
3	13400000000	22020000	608,5	55,53
4	14310000000	21410000	668,5	56,56
5	7656000000	3480000	2200	35,53
6	11230000000	21560000	520,9	34,96
7	12350000000	22030000	560,5	53,89
8	13400000000	22020000	608,5	55,53
9	14310000000	21410000	668,5	56,56
10	7656000000	3480000	2200	35,53
11	11230000000	21560000	520,9	34,96
12	12350000000	22030000	560,5	53,89
13	13400000000	22020000	608,5	55,53
14	14310000000	21410000	668,5	56,56
15	7656000000	3480000	2200	35,53
16	11230000000	21560000	520,9	34,96
17	12350000000	22030000	560,5	53,89
18	13400000000	21410000	608,5	55,53
19	11230000000	21560000	520,9	34,96
20	12350000000	22030000	560,5	53,89
21	13400000000	22020000	608,5	55,53
22	14310000000	21410000	668,5	56,56
23	7656000000	3480000	2200	35,53
24	14310000000	21410000	668,5	56,56
25	12350000000	22030000	560,5	53,89
26	13400000000	22020000	608,5	55,53
27	14310000000	21410000	668,5	56,56
28	7656000000	3480000	2200	35,53
29	14310000000	21410000	668,5	56,56
30	12350000000	22030000	560,5	53,89
31	7656000000	3480000	2200	35,53
32	11230000000	21560000	520,9	34,96
33	12350000000	22030000	560,5	53,89
34	13400000000	22020000	608,5	55,53
35	14310000000	21410000	668,5	56,56
36	7656000000	3480000	2200	35,53
37	11230000000	21560000	520,9	34,96
38	12350000000	22030000	560,5	53,89
39	13400000000	22020000	608,5	55,53
40	14310000000	21410000	668,5	56,56
41	7656000000	3480000	2200	35,53
42	11230000000	21560000	520,9	34,96
43	12350000000	22030000	560,5	53,89
44	13400000000	22020000	608,5	55,53
45	14310000000	21410000	668,5	56,56

46	12350000000	22030000	560,5	53,89
47	13400000000	22020000	608,5	55,53
48	14310000000	21410000	668,5	56,56
49	7656000000	3480000	2200	35,53
50	13400000000	22020000	608,5	55,53
51	14310000000	21410000	668,5	56,56
52	7656000000	3480000	2200	35,53
53	14310000000	21410000	668,5	56,56
54	12350000000	22030000	560,5	53,89
55	13400000000	22020000	608,5	55,53
56	14310000000	21410000	668,5	56,56
57	14310000000	21410000	668,5	56,56
58	11230000000	21560000	520,9	34,96
59	12350000000	22030000	560,5	53,89
60	13400000000	22020000	608,5	55,53
61	14310000000	21410000	668,5	56,56
62	7656000000	3480000	2200	35,53
63	14310000000	21410000	668,5	56,56
64	12350000000	22030000	560,5	53,89
65	13400000000	22020000	608,5	55,53
66	11230000000	21560000	520,9	34,96
67	14310000000	21410000	668,5	56,56
68	14310000000	21410000	668,5	56,56
69	13400000000	22020000	608,5	55,53
70	13400000000	22020000	608,5	55,53
71	13400000000	22020000	608,5	55,53
72	7656000000	3480000	2200	35,53
73	12350000000	22030000	560,5	53,89
74	12350000000	22030000	560,5	53,89
75	13400000000	22020000	608,5	55,53
76	14310000000	21410000	668,5	56,56
77	14310000000	21410000	668,5	56,56
78	14310000000	21410000	668,5	34,96
79	12350000000	22030000	560,5	53,89
80	13400000000	22020000	608,5	55,53
81	13400000000	22020000	608,5	55,53
82	14310000000	21410000	668,5	56,56
83	12350000000	22030000	560,5	53,89
84	13400000000	22020000	608,5	55,53
85	13400000000	22020000	608,5	55,53
86	12350000000	22030000	560,5	53,89
87	14310000000	21410000	668,5	56,56
88	12350000000	22030000	560,5	53,89
89	13400000000	22020000	608,5	55,53
Média	**12,36 Gflops**			

Plataforma 3				
Máquinas	**Poder (*flops*)**	***flops/watt***	***Watts***	**Cons. ocioso/hora**
0	4712500000	3250000	1450	102,1
1	7656000000	3480000	2200	35,53
2	11230000000	21560000	520,9	34,96
3	12350000000	22030000	560,5	53,89
4	13400000000	22020000	608,5	55,53
5	14310000000	21410000	668,5	56,56
6	19550000000	27410000	713,2	20,38
7	19964000000	7130000	2800	34,57
8	20311000000	10690000	1900	49
9	4712500000	3250000	1450	102,1
10	7656000000	3480000	2200	35,53
11	11230000000	21560000	520,9	34,96
12	11230000000	21560000	520,9	34,96
13	11230000000	21560000	520,9	34,96
14	14310000000	21410000	668,5	56,56
15	19550000000	27410000	713,2	20,38
16	19964000000	7130000	2800	34,57
17	20311000000	10690000	1900	49
18	19550000000	27410000	713,2	20,38
19	19964000000	7130000	2800	34,57
20	20311000000	10690000	1900	49
21	4712500000	3250000	1450	102,1
22	7656000000	3480000	2200	35,53
23	11230000000	21560000	520,9	34,96
24	19964000000	7130000	2800	34,57
25	19964000000	7130000	2800	34,57
26	4712500000	3250000	1450	102,1
27	7656000000	3480000	2200	35,53
28	11230000000	21560000	520,9	34,96
29	12350000000	22030000	560,5	53,89
30	13400000000	22020000	608,5	55,53
31	14310000000	21410000	668,5	56,56
32	4712500000	3250000	1450	102,1
33	19964000000	7130000	2800	34,57
34	19550000000	27410000	713,2	20,38
35	19964000000	7130000	2800	34,57
36	20311000000	10690000	1900	49
37	4712500000	3250000	1450	102,1
38	7656000000	3480000	2200	35,53
39	11230000000	21560000	520,9	34,96
40	19964000000	7130000	2800	34,57
41	20311000000	10690000	1900	49
42	4712500000	3250000	1450	102,1
43	4712500000	3250000	1450	102,1
44	7656000000	3480000	2200	35,53
45	11230000000	21560000	520,9	34,96

46	19964000000	7130000	2800	34,57
47	20311000000	10690000	2800	34,57
48	4712500000	3250000	1450	102,1
49	7656000000	3480000	2200	35,53
50	11230000000	21560000	520,9	34,96
51	7656000000	3480000	2200	35,53
52	11230000000	21560000	520,9	34,96
53	11230000000	21560000	520,9	34,96
54	19550000000	27410000	713,2	20,38
55	19964000000	7130000	2800	34,57
56	4712500000	3250000	1450	102,1
57	19964000000	7130000	2800	34,57
58	20311000000	10690000	1900	49
59	4712500000	3250000	1450	102,1
60	4712500000	3250000	1450	102,1
61	7656000000	3480000	2200	35,53
62	11230000000	21560000	520,9	34,96
63	11230000000	21560000	520,9	34,96
64	20311000000	10690000	1900	49
65	4712500000	3250000	1450	102,1
66	7656000000	3480000	2200	35,53
67	11230000000	21560000	520,9	34,96
68	4712500000	3250000	1450	102,1
69	19550000000	27410000	713,2	20,38
70	7656000000	3480000	2200	35,53
71	19964000000	7130000	2800	34,57
72	4712500000	3250000	1450	102,1
73	4712500000	3250000	1450	102,1
74	4712500000	3250000	1450	102,1
75	19964000000	7130000	2800	34,57
76	4712500000	3250000	1450	102,1
77	11230000000	21560000	520,9	34,96
78	19964000000	7130000	2800	34,57
79	19550000000	27410000	713,2	20,38
80	11230000000	21560000	520,9	34,96
81	7656000000	3480000	2200	35,53
82	19964000000	7130000	2800	34,57
83	7656000000	3480000	2200	35,53
84	4712500000	3250000	1450	102,1
85	7656000000	3480000	2200	35,53
86	7656000000	3480000	2200	35,53
87	4712500000	3250000	1450	102,1
88	11230000000	21560000	520,9	34,96
89	19550000000	27410000	713,2	20,3832
Média	**12,36 Gflops**			

APÊNDICE B – CONFIGURAÇÃO DE CONSUMO DOS *LINKS*

Origem	Destino	Consumo do *link*																			Consumo Total
Tremblay	Jacquelin	1,7	4,1	3,7	2,1	13,7	8,75	2,1	3,7	4,1	1,7	8,75									72,4
	Descr. Link	17	72	65	64	69	3	2	54	56	59	145									
Tremblay	Boivin	1,7	4,1	3,7	2,1	13,7	8,75	2,1	3,7	4,1	8,75	2,1	13,7								77,5
	Descr. Link	17	72	65	64	69	3	0	16	1	11	44	47								
Tremblay	Jean_Yves	1,7	4,1	3,7	2,1	13,7	8,75	2,1	3,7	4,1	3,7	13,7	4,1								74,45
	Descr. Link	17	72	65	64	69	3	2	54	56	58	63	138								
Tremblay	TeX	1,7	4,1	3,7	2,1	13,7	8,75	2,1	3,7	4,1	8,75	4,1	13,7	4,1	2,1	1,7	4,1	3,7			14,2
	Descr. Link	17	72	65	64	69	3	0	16	1	6	7	19	28	27	3	35	1			
Tremblay	Geoff	1,7	4,1	3,7	2,1	13,7	8,75	2,1	3,7	4,1	1,7										63,65
	Descr. Link	17	72	65	64	69	3	2	54	56	59										
Tremblay	Disney	1,7	4,1	3,7	2,1	13,7	8,75	2,1	3,7	4,1	8,75	4,1	13,7	4,1	2,1	1,7	1,7	4,1	13,7		124,9
	Descr. Link	17	72	65	64	69	3	0	16	1	6	7	19	28	27	3	32	33	111		
Tremblay	iRMX	1,7	4,1	3,7	2,1	13,7	8,75	2,1	3,7	4,1	8,75	2,1	3,7	4,1	8,75	1,7	3,7	8,75	3,7	13,7	12,9
	Descr. Link	17	72	65	64	69	3	0	16	1	6	9	79	78	8	82	87	88	89	172	
Tremblay	McGee	1,7	4,1	3,7	2,1	13,7	8,75	2,1	3,7	4,1	8,75	3,7	2,1	4,1	13,7						85,3
	Descr. Link	17	72	65	64	69	3	0	16	1	6	8	48	38	41						
Tremblay	Gatien	1,7	4,1	3,7	2,1	13,7	8,75	2,1	3,7	4,1	8,75	3,7	2,1	4,1	3,7	8,75					84,5
	Descr. Link	17	72	65	64	69	3	0	16	1	6	8	48	38	4	115					
Tremblay	Laroche	1,7	4,1	3,7	2,1	13,7	8,75	2,1	3,7	4,1	8,75	2,1	3,7	4,1	8,75	1,7	3,7				94,75
	Descr. Link	17	72	65	64	69	3	0	16	1	6	9	79	78	8	82	91				
Tremblay	Tanguay	1,7	4,1	3,7	2,1	13,7	2,1	8,75													45,15
	Descr. Link	17	72	65	64	69	7	162													
Tremblay	Morin	1,7	4,1	3,7	2,1	13,7	8,75	2,1	3,7	4,1	8,75	4,1	13,7	4,1	2,1	3,7	13,7				13,1
	Descr. Link	17	72	65	64	69	3	0	16	1	6	7	19	28	27	31	16				
Tremblay	Ethernet	1,7	4,1	3,7	2,1	13,7	8,75	2,1	3,7	4,1	8,75	2,1	13,7	3,7							81,2
	Descr. Link	17	72	65	64	69	3	0	16	1	11	44	47	118							
Tremblay	Bellemarre	1,7	4,1	3,7	2,1	13,7	8,75	2,1	3,7	4,1	8,75	4,1	13,7	4,1	2,1	1,7	3,7				1,1
	Descr. Link	17	72	65	64	69	3	0	16	1	6	7	19	28	27	3	34				
Tremblay	Kuenning	1,7	4,1	3,7	2,1	13,7	8,75	2,1	3,7	4,1	3,7	1,7	8,75								76,1
	Descr. Link	17	72	65	64	69	3	2	54	56	58	62	135								
Tremblay	Gaston	1,7	4,1	3,7	2,1	8,75	2,1	3,7													35,15
	Descr. Link	17	72	65	64	73	75	153													
Tremblay	Harry	1,7	4,1	3,7	2,1	13,7	8,75	2,1	3,7	4,1	8,75	4,1	13,7	4,1	2,1	1,7	1,7	4,1	3,7		114,9
	Descr. Link	17	72	65	64	69	3	0	16	1	6	7	19	28	27	3	32	33	18		
Tremblay	Bousquet	1,7	4,1	3,7	2,1	13,7	8,75	2,1	3,7	4,1	8,75	3,7	2,1	4,1	1,7	2,1	13,7				98,1
	Descr. Link	17	72	65	64	69	3	0	16	1	6	8	48	38	36	37	124				
Tremblay	Paul	1,7	4,1	3,7	2,1	13,7	8,75	2,1	3,7	4,1	8,75	4,1	13,7	4,1	13,7	8,75					16,5
	Descr. Link	17	72	65	64	69	3	0	16	1	6	7	19	28	29	93					
Tremblay	St_Jean	1,7	4,1	3,7	2,1	13,7	8,75	2,1	3,7	4,1	8,75	4,1	13,7	4,1	2,1	1,7	3,7	13,7			113,8
	Descr. Link	17	72	65	64	69	3	0	16	1	6	7	19	28	27	3	34	97			
Tremblay	Jill	1,7	4,1	3,7	2,1	13,7	8,75	2,1	3,7	4,1	8,75	2,1	3,7	4,1	8,75	1,7	3,7	8,75	2,1		15,6
	Descr. Link	17	72	65	64	69	3	0	16	1	6	9	79	78	8	82	87	88	9		
Tremblay	Marcel	1,7	4,1	3,7	2,1	8,75	2,1	3,7													35,15
	Descr. Link	17	72	65	64	73	75	155													
Tremblay	Wright	1,7	4,1	3,7	2,1	13,7	8,75	2,1	3,7	4,1	1,7	4,1									67,75
	Descr. Link	17	72	65	64	69	3	2	54	56	59	147									
Tremblay	Jackson	1,7	4,1	3,7	2,1	13,7	8,75	2,1	3,7	4,1	3,7	1,7									67,35
	Descr. Link	17	72	65	64	69	3	2	54	56	58	62									
Tremblay	Fernand	1,7	4,1	3,7	2,1	13,7	8,75	2,1	3,7	4,1	8,75	4,1	13,7	4,1	2,1	1,7	1,7	4,1	8,75		119,95
	Descr. Link	17	72	65	64	69	3	0	16	1	6	7	19	28	27	3	32	33	19		
Tremblay	Soucy	1,7	4,1	3,7	2,1	13,7	8,75	2,1	3,7	4,1	8,75	2,1	3,7	4,1	8,75	1,7	3,7				94,75
	Descr. Link	17	72	65	64	69	3	0	16	1	6	9	79	78	8	83	191				
Tremblay	Ste_Julie	1,7	4,1	3,7	2,1	8,75	2,1														31,45
	Descr. Link	17	72	65	64	73	75														
Tremblay	Victoria	1,7	4,1	3,7	2,1	13,7	8,75	2,1	3,7	4,1	3,7	1,7	1,7								78,5
	Descr. Link	17	72	65	64	69	3	2	54	56	58	62	133								
Tremblay	Lessard	1,7	4,1	3,7	2,1	13,7	8,75	2,1	3,7	4,1	8,75	4,1	13,7	4,1	2,1	1,7	3,7	2,1			12,2
	Descr. Link	17	72	65	64	69	3	0	16	1	6	7	19	28	27	3	34	99			
Tremblay	Florient	1,7	4,1	3,7	2,1	13,7	8,75	2,1	3,7	4,1	8,75	2,1	3,7	4,1	4,1	8,75					84,45
	Descr. Link	17	72	65	64	69	3	0	16	1	6	9	79	78	81	181					

Tremblay	Marcoux	1,7	4,1	3,7	2,1	13,7	8,75	2,1	3,7	4,1	8,75	2,1	3,7	4,1	8,75	1,7	3,7	3,7			98,45
	Descr. Link	17	72	65	64	69	3	0	16	1	6	9	79	78	8	82	91	186			
Tremblay	Gavrel	1,7	4,1	3,7	2,1	13,7	8,75	2,1	2,1	1,7	2,1										6,5
	Descr. Link	17	72	65	64	69	3	2	52	53	148										
Tremblay	Bescherelle	1,7	4,1	3,7	2,1	13,7	8,75	2,1	3,7	3,7	13,7										66,25
	Descr. Link	17	72	65	64	69	3	2	54	57	14										
Tremblay	Pierre	1,7	4,1	3,7	2,1	8,75	2,1	3,7	13,7												48,85
	Descr. Link	17	72	65	64	73	74	76	167												
Tremblay	Jamie	1,7	4,1	3,7	2,1	13,7	8,75	2,1	3,7	4,1	8,75	3,7	2,1	4,1	13,7	1,7					96
	Descr. Link	17	72	65	64	69	3	0	16	1	6	8	48	38	41	12					
Tremblay	Rubin	1,7	4,1	3,7	2,1	13,7	8,75	2,1	3,7	4,1	1,7	4,1									67,75
	Descr. Link	17	72	65	64	69	3	2	54	56	59	144									
Tremblay	Olivier	1,7	4,1																		14,8
	Descr. Link	17	171																		
Tremblay	Boucherville	1,7	4,1	3,7	2,1	13,7	8,75	2,1	3,7	4,1	8,75	2,1	3,7	4,1	8,75	1,7	3,7	8,75	3,7		17,2
	Descr. Link	17	72	65	64	69	3	0	16	1	6	9	79	78	8	82	87	88	89		
Tremblay	Pointe_Claire	1,7	4,1	3,7	2,1	13,7	8,75	2,1	3,7	4,1	8,75	2,1	3,7	4,1	8,75	1,7	3,7	3,7			98,45
	Descr. Link	17	72	65	64	69	3	0	16	1	6	9	79	78	8	82	91	184			
Tremblay	Kansas	1,7	4,1	3,7	2,1	13,7	8,75	2,1	3,7	4,1	8,75	3,7	2,1	4,1	13,7	13,7					99
	Descr. Link	17	72	65	64	69	3	0	16	1	6	8	48	38	41	122					
Tremblay	King	1,7	4,1	3,7	2,1	13,7	8,75	2,1	3,7	4,1	3,7	13,7	2,1								72,45
	Descr. Link	17	72	65	64	69	3	2	54	56	58	63	137								
Tremblay	Lapointe	1,7	4,1	3,7	2,1	13,7	8,75	2,1	3,7	4,1	8,75	2,1	3,7	4,1	4,1						75,7
	Descr. Link	17	72	65	64	69	3	0	16	1	6	9	79	78	81						
Tremblay	Julian	1,7	4,1	3,7	2,1	13,7	8,75	2,1	3,7	4,1	8,75	4,1	13,7	4,1	2,1	1,7	1,7	4,1			111,2
	Descr. Link	17	72	65	64	69	3	0	16	1	6	7	19	28	27	3	32	33			
Tremblay	Lafontaine	1,7	4,1	3,7	2,1	13,7	8,75	2,1	3,7	4,1	3,7	13,7	1,7								81,5
	Descr. Link	17	72	65	64	69	3	2	54	56	58	63	139								
Tremblay	Gordon	1,7	4,1	3,7	2,1	13,7	8,75	2,1	3,7	4,1	8,75	2,1	3,7	4,1	8,75	1,7					91,5
	Descr. Link	17	72	65	64	69	3	0	16	1	6	9	79	78	8	83					
Tremblay	Drouin	1,7	4,1	3,7	2,1	13,7	8,75	2,1	3,7	4,1	8,75	3,7	2,1	4,1	13,7	2,1					87,4
	Descr. Link	17	72	65	64	69	3	0	16	1	6	8	48	38	41	123					
Tremblay	Robert	1,7	4,1	3,7	2,1	13,7	8,75	2,1	3,7	4,1	8,75	2,1	3,7	4,1	8,75	1,7	3,7	3,7			98,45
	Descr. Link	17	72	65	64	69	3	0	16	1	6	9	79	78	8	82	91	187			
Tremblay	Jocelyne	1,7	4,1	3,7	2,1	13,7	8,75	2,1	3,7	4,1	8,75	2,1	3,7	4,1	8,75	3,7	8,75	3,7	1,7		17,2
	Descr. Link	17	72	65	64	69	3	0	16	1	6	9	79	78	8	87	88	89	173		
Tremblay	Stephen	1,7	4,1	3,7	2,1	13,7	8,75	2,1	3,7	4,1	8,75	4,1	13,7	4,1	2,1	1,7	4,1	4,1			14,6
	Descr. Link	17	72	65	64	69	3	0	16	1	6	7	19	28	27	3	35	11			
Tremblay	Provost	1,7	4,1	3,7	2,1	13,7	8,75	2,1	3,7	4,1	8,75	4,1	13,7	4,1	2,1	1,7	4,1	13,7			114,2
	Descr. Link	17	72	65	64	69	3	0	16	1	6	7	19	28	27	3	35	13			
Tremblay	Juneau	1,7	4,1	3,7	2,1	13,7	8,75	2,1	3,7	4,1	8,75	2,1	3,7	4,1	8,75	1,7	4,1				95,15
	Descr. Link	17	72	65	64	69	3	0	16	1	6	9	79	78	8	83	189				
Tremblay	Casavant	1,7	4,1	3,7	2,1	13,7	8,75	2,1	3,7	4,1	8,75	4,1	13,7	4,1	13,7	8,75					16,5
	Descr. Link	17	72	65	64	69	3	0	16	1	6	7	19	28	29	92					
Tremblay	St_Antoine	1,7	4,1	3,7	2,1	13,7	8,75	2,1	3,7	4,1	3,7	13,7									7,35
	Descr. Link	17	72	65	64	69	3	2	54	56	58	63									
Tremblay	Louis	1,7	4,1	3,7	2,1	13,7	8,75	2,1	3,7	4,1	8,75	2,1	13,7	2,1							79,6
	Descr. Link	17	72	65	64	69	3	0	16	1	11	44	47	117							
Tremblay	Julien	1,7	4,1	3,7	2,1	13,7	8,75	2,1	3,7	4,1	3,7	1,7	3,7								71,5
	Descr. Link	17	72	65	64	69	3	2	54	56	58	62	134								
Tremblay	St_Paul	1,7	4,1	3,7	2,1	8,75	2,1	3,7	13,7												48,85
	Descr. Link	17	72	65	64	73	74	76	165												
Tremblay	Mathematica	1,7	4,1	3,7	2,1	1,7															31,3
	Descr. Link	17	72	65	66	157															
Tremblay	LaTeX	1,7	4,1	3,7	2,1	13,7	8,75	2,1	3,7	4,1	8,75	2,1	3,7	4,1	8,75	1,7	3,7	8,75	2,1	8,75	114,35
	Descr. Link	17	72	65	64	69	3	0	16	1	6	9	79	78	8	82	87	88	9	177	
Tremblay	Sirois	1,7	4,1	3,7	2,1	13,7	8,75	2,1	3,7	4,1	8,75	3,7	13,7	1,7	4,1						93,9
	Descr. Link	17	72	65	64	69	3	0	16	1	6	8	42	43	129						
Tremblay	Monique	1,7	4,1	3,7	2,1	13,7	8,75	2,1	3,7	4,1	8,75	3,7	2,1	4,1	3,7	2,1					77,4
	Descr. Link	17	72	65	64	69	3	0	16	1	6	8	48	38	4	113					

Tremblay	Audy	1,7	4,1	3,7	2,1	13,7	8,75	2,1	3,7	4,1	8,75	2,1	3,7	4,1	4,1	3,7					79,4
	Descr. Link	17	72	65	64	69	3	0	16	1	6	9	79	78	81	183					
Tremblay	Bourassa	1,7	4,1	3,7	2,1	13,7	8,75	2,1	2,1	1,7	3,7										61,65
	Descr. Link	17	72	65	64	69	3	2	52	53	15										
Tremblay	OHara	1,7	4,1	3,7	2,1	13,7	2,1	4,1													4,5
	Descr. Link	17	72	65	64	69	7	163													
Tremblay	Boston	1,7	4,1	3,7	2,1	13,7	8,75	2,1	3,7	4,1	8,75	4,1	13,7	4,1	13,7	3,7					11
	Descr. Link	17	72	65	64	69	3	0	16	1	6	7	19	28	29	95					
Tremblay	Papineau	1,7	4,1	3,7	2,1	13,7	8,75	2,1	3,7	4,1	8,75	2,1	3,7	4,1	4,1	2,1					77,8
	Descr. Link	17	72	65	64	69	3	0	16	1	6	9	79	78	81	182					
Tremblay	Fraser	1,7	4,1	3,7	2,1	13,7	8,75	2,1	3,7	4,1	8,75	4,1	13,7	4,1	2,1	1,7	3,7	2,1			12,2
	Descr. Link	17	72	65	64	69	3	0	16	1	6	7	19	28	27	3	34	96			
Tremblay	Browne	1,7	4,1	3,7	2,1	13,7	8,75	2,1	1,7												55,85
	Descr. Link	17	72	65	64	69	73	75	152												
Tremblay	SunOS	1,7	4,1	3,7	2,1	13,7	8,75	2,1	3,7	4,1	8,75	3,7	2,1	4,1	3,7	13,7					89
	Descr. Link	17	72	65	64	69	3	0	16	1	6	8	48	38	4	114					
Tremblay	Varennes	1,7	4,1	3,7	2,1	13,7	8,75	2,1	3,7	4,1	3,7	13,7	13,7								84,5
	Descr. Link	17	72	65	64	69	3	2	54	56	58	63	136								
Tremblay	Jacques	1,7	4,1	3,7	2,1	3,7															24,3
	Descr. Link	17	72	65	66	159															
Tremblay	Thierry	1,7	4,1	3,7	2,1	4,1															24,7
	Descr. Link	17	72	65	66	156															
Tremblay	Lachapelle	1,7	4,1	3,7	2,1	13,7	8,75	2,1	3,7	4,1	1,7	1,7									74,35
	Descr. Link	17	72	65	64	69	3	2	54	56	59	146									
Tremblay	kV	1,7	4,1	3,7	2,1	13,7	8,75	2,1	3,7	3,7											52,55
	Descr. Link	17	72	65	64	69	3	2	54	57											
Tremblay	Yolande	1,7	13,7																		24,4
	Descr. Link	17	168																		
Tremblay	Pronovost	1,7	4,1	3,7	2,1	13,7	8,75	2,1	2,1	1,7	4,1										62,5
	Descr. Link	17	72	65	64	69	3	2	52	53	151										
Tremblay	SPICE	1,7	4,1	3,7	2,1	13,7	8,75	2,1	3,7	4,1	8,75	4,1	13,7	4,1	2,1	1,7	1,7	4,1	3,7		114,9
	Descr. Link	17	72	65	64	69	3	0	16	1	6	7	19	28	27	3	32	33	11		
Tremblay	Europe	1,7	4,1	3,7	2,1	13,7	8,75	2,1	3,7	4,1	8,75	2,1	3,7	4,1	8,75	1,7	3,7	13,7			18,45
	Descr. Link	17	72	65	64	69	3	0	16	1	6	9	79	78	8	82	91	185			
Tremblay	Roy	1,7	4,1	3,7	2,1	8,75	2,1	3,7													35,15
	Descr. Link	17	72	65	64	73	74	76													
Tremblay	Amadeus	1,7	4,1	3,7	2,1	13,7	8,75	2,1	2,1	1,7											57,95
	Descr. Link	17	72	65	64	69	3	2	52	53											
Tremblay	Intel	1,7	4,1	3,7	2,1	13,7	8,75	2,1	3,7	3,7	3,7										56,25
	Descr. Link	17	72	65	64	69	3	2	54	57	141										
Tremblay	Cambridge	1,7	4,1	3,7	2,1	8,75															29,35
	Descr. Link	17	72	65	66	158															
Tremblay	UNIX	1,7	4,1	3,7	2,1	13,7	8,75	2,1	3,7	4,1	8,75	3,7	13,7	1,7							89,8
	Descr. Link	17	72	65	64	69	3	0	16	1	6	8	42	43							
Tremblay	Domey	1,7	4,1	3,7	2,1	13,7	8,75	2,1	3,7	4,1	8,75	3,7	2,1	4,1	3,7						75,3
	Descr. Link	17	72	65	64	69	3	0	16	1	6	8	48	38	4						
Tremblay	St_Bruno	1,7	4,1	3,7	2,1	13,7	8,75	2,1	3,7	4,1	8,75	2,1	13,7	4,1							81,6
	Descr. Link	17	72	65	64	69	3	0	16	1	11	44	47	119							
Tremblay	Jean_Claude	1,7	4,1	3,7	2,1	13,7	2,1	2,1													38,5
	Descr. Link	17	72	65	64	69	7	16													
Tremblay	Doyon	1,7	4,1	3,7	2,1	13,7	8,75	2,1	3,7	4,1	3,7	1,7	13,7								81,5
	Descr. Link	17	72	65	64	69	3	2	54	56	58	62	132								
Tremblay	Fafard	1,7	4,1	3,7	2,1	13,7	8,75	2,1	3,7	4,1	8,75	2,1	3,7	4,1	8,75	1,7	3,7	8,75	2,1	8,75	114,35
	Descr. Link	17	72	65	64	69	3	0	16	1	6	9	79	78	8	82	87	88	9	179	
Tremblay	Croteau	1,7	4,1	3,7	2,1	13,7	8,75	2,1	3,7	4,1	8,75	2,1	3,7	4,1	8,75	1,7	3,7	8,75	3,7	2,1	19,3
	Descr. Link	17	72	65	64	69	3	0	16	1	6	9	79	78	8	82	87	88	89	175	
Tremblay	Dodge	1,7	4,1	3,7	2,1	13,7	8,75	2,1	3,7	4,1	8,75	2,1	3,7	4,1	4,1						75,7
	Descr. Link	17	72	65	64	69	3	0	16	1	6	9	79	78	81						

APÊNDICE C – RESULTADOS DOS TESTES

Plataforma 1 – 720 tarefas

0 %

Algoritmos	Tempo (seg)	Tempo (horas)	Consumo (watts)	Cons. Ocioso (watts)	CONSUMO TOTAL	Rede (watts)	C. T. Mega Watts
WQ	7004,05	1,95	328965699,98	265,05	**332896091,20**	**3930126,16**	**332,90**
WQR	7096,23	1,97	336236563,26	679,92	**340167369,77**	**3930126,59**	**340,17**
Suff	7294,15	2,03	336751627,99	1634,18	**340683632,11**	**3930369,94**	**340,68**
XSuff	9099,20	2,53	329256378,55	3141,60	**332753256,72**	**3493736,57**	**332,75**
DFPLTF	7285,83	2,02	329256378,55	992,59	**332751066,10**	**3493694,95**	**332,75**
LECSA	26873,19	7,46	326827054,02	147,10	**326827207,12**	**6,00**	**326,83**
LECSA2	26873,19	7,46	326827054,02	147,10	**326827207,12**	**6,00**	**326,83**
LECSA3	24794,92	6,89	326975476,84	144,79	**326975629,51**	**7,88**	**326,98**
DLECSA	26873,19	7,46	326827054,02	147,10	**326827207,12**	**6,00**	**326,83**
DLECSA2	26873,19	7,46	326827054,02	147,10	**326827207,12**	**6,00**	**326,83**
DLECSA3	24794,92	6,89	326975476,84	144,79	**326975629,51**	**7,88**	**326,98**
ECI	10852,29	3,01	327024958,00	4569,72	**331833326,47**	**4803798,75**	**331,83**
EWQ	7032,05	1,95	328965699,98	281,28	**332896107,89**	**3930126,63**	**332,90**
ESUFF	7385,91	2,05	338110418,31	2590,81	**341606817,28**	**3493808,16**	**341,61**
EXSUFF	9099,20	2,53	329256378,55	3141,60	**332753256,72**	**3493736,57**	**332,75**
EDFPLTF	7285,83	2,02	329256378,55	992,59	**332751178,88**	**3493807,73**	**332,75**

25%

Algoritmos	Tempo (seg)	Tempo (horas)	Consumo (watts)	Cons. Ocioso (watts)	CONSUMO TOTAL	Rede (watts)	C. T. Mega Watts
WQ	7270,22	2,02	329047604,32	261,67	**332685420,13**	**3637554,14**	**332,69**
WQR	7361,36	2,04	330308285,23	1022,88	**333946862,30**	**3637554,19**	**333,95**
Suff	9285,21	2,58	386349674,53	4047,07	**387773212,54**	**1419490,94**	**387,77**
XSuff	9928,61	2,76	329261327,32	3423,09	**332837088,72**	**3572338,32**	**332,84**
DFPLTF	7572,50	2,10	329199297,34	1167,60	**332216034,03**	**3015569,09**	**332,22**
LECSA	30190,82	8,39	326827054,17	1241,80	**326828301,86**	**5,88**	**326,83**
LECSA2	30190,82	8,39	326827054,02	1241,80	**326828301,70**	**5,88**	**326,83**
LECSA3	27873,28	7,74	326975476,84	1189,43	**326976673,99**	**7,71**	**326,98**
DLECSA	33419,68	9,28	326827054,02	2305,36	**326829365,11**	**5,74**	**326,83**
DLECSA2	33419,68	9,28	326827054,02	2305,36	**326829365,11**	**5,74**	**326,83**
DLECSA3	31351,24	8,71	326975476,84	2370,11	**326977854,36**	**7,41**	**326,98**
ECI	11383,57	3,16	327031680,56	5217,03	**331539397,84**	**4502500,25**	**331,54**
EWQ	7103,10	1,97	328984361,17	242,11	**332871062,53**	**3886459,24**	**332,87**
ESUFF	8625,30	2,40	352150050,88	2695,33	**356137820,11**	**3985073,91**	**356,14**
EXSUFF	10124,61	2,81	329261327,32	4392,30	**332838057,84**	**3572338,23**	**332,84**
EDFPLTF	7566,23	2,10	329229644,38	1274,57	**332794641,54**	**3563722,60**	**332,79**

50%							
Algoritmos	**Tempo (seg)**	**Tempo (horas)**	**Consumo (watts)**	**Cons.Ocioso (watts)**	**CONSUMO TOTAL**	**Rede (watts)**	**C. T. Mega Watts**
WQ	7370,25	2,05	329050038,00	260,88	**332652919,02**	**3602620,15**	**332,65**
WQR	7461,83	2,07	329629815,36	1143,78	**333233579,32**	**3602620,18**	**333,23**
Suff	8912,75	2,48	369066577,97	3116,81	**370925854,83**	**1856160,05**	**370,93**
XSuff	12534,96	3,48	329262047,83	7239,71	**332391842,30**	**3122554,76**	**332,39**
DFPLTF	7705,32	2,14	329177503,67	1376,18	**329526072,46**	**347192,61**	**329,53**
LECSA	16315,96	4,53	326875662,49	2534,60	**326878226,79**	**29,70**	**326,88**
LECSA2	15047,75	4,18	326898204,21	1739,31	**326899973,62**	**30,11**	**326,90**
LECSA3	16315,95	4,53	326926002,57	2795,12	**330072885,57**	**3144087,88**	**330,07**
DLECSA	17620,16	4,89	326886557,14	3442,90	**326890027,97**	**27,93**	**326,89**
DLECSA2	17344,18	4,82	326889206,07	3258,28	**326892492,15**	**27,80**	**326,89**
DLECSA3	17102,01	4,75	326940020,28	3324,71	**330393102,92**	**3449757,94**	**330,39**
ECI	14314,63	3,98	327064003,65	8754,94	**332618907,57**	**5546148,98**	**332,62**
EWQ	7297,35	2,03	329039287,05	241,69	**332882320,64**	**3842791,90**	**332,88**
ESUFF	8388,67	2,33	346944558,80	2113,93	**351073665,12**	**4126992,40**	**351,07**
EXSUFF	12250,85	3,40	329262047,83	6345,44	**332783955,39**	**3515562,12**	**332,78**
EDFPLTF	7775,59	2,16	329267016,13	1586,90	**333002676,60**	**3734073,57**	**333,00**

75%							
Algoritmos	**Tempo (seg)**	**Tempo (horas)**	**Consumo (watts)**	**Cons. Ocioso (watts)**	**CONSUMO TOTAL**	**Rede (watts)**	**C. T. Mega Watts**
WQ	7400,51	2,06	329044239,93	260,66	**332799955,57**	**3755454,99**	**332,80**
WQR	7491,39	2,08	330175743,13	1175,89	**333932374,05**	**3755455,04**	**333,93**
Suff	9420,96	2,62	382402256,00	3945,84	**384786360,63**	**2380158,78**	**384,79**
XSuff	15631,97	4,34	329263841,30	10762,21	**332626401,92**	**3351798,41**	**332,63**
DFPLTF	7442,40	2,07	329234979,68	1144,43	**332325994,93**	**3089870,81**	**332,33**
LECSA	12353,82	3,43	327328496,07	3701,98	**329607840,38**	**2275642,32**	**329,61**
LECSA2	11400,36	3,17	327356315,05	2850,76	**330983889,44**	**3624723,64**	**330,98**
LECSA3	12359,49	3,43	327328496,07	3705,64	**330956927,10**	**3624725,39**	**330,96**
DLECSA	11839,67	3,29	327533639,88	3277,59	**331202266,68**	**3665349,21**	**331,20**
DLECSA2	12520,74	3,48	327515750,69	3967,10	**331282589,50**	**3762871,70**	**331,28**
DLECSA3	12697,23	3,53	327556760,18	3712,62	**331331475,50**	**3771002,71**	**331,33**
ECI	14897,11	4,14	327068477,15	9442,73	**332733232,35**	**5655312,46**	**332,73**
EWQ	7398,58	2,06	329035710,28	241,39	**332682238,52**	**3646286,85**	**332,68**
ESUFF	8963,22	2,49	355328851,79	3177,67	**359273429,22**	**3941399,76**	**359,27**
EXSUFF	16110,17	4,48	329263841,30	10853,37	**332080649,63**	**2805954,96**	**332,08**
EDFPLTF	7659,46	2,13	329265753,44	1434,14	**332553716,81**	**3286529,23**	**332,55**

100%							
Algoritmos	**Tempo (seg)**	**Tempo (horas)**	**Consumo (watts)**	**Cons. Ocioso (watts)**	**CONSUMO TOTAL**	**Rede (watts)**	**C. T. Mega Watts**
WQ	7331,21	2,04	329050470,90	260,19	**332937189,42**	**3886458,33**	**332,94**
WQR	7421,73	2,06	330537420,86	1096,91	**334424976,17**	**3886458,40**	**334,42**
Suff	9018,47	2,51	364537966,28	3194,16	**367816501,65**	**3275341,21**	**367,82**
XSuff	19135,96	5,32	329229186,65	14640,94	**332999548,63**	**3755721,04**	**333,00**
DFPLTF	7585,35	2,11	329211519,24	1315,87	**332662996,82**	**3450161,71**	**332,66**
LECSA	9823,67	2,73	328270484,66	3692,21	**331418781,36**	**3144604,48**	**331,42**
LECSA2	9062,68	2,52	328293613,89	2846,36	**332488924,89**	**4192464,65**	**332,49**
LECSA3	9823,67	2,73	328270484,66	3690,88	**332466984,43**	**4192808,89**	**332,47**
DLECSA	10189,59	2,83	328965535,24	4338,57	**332507357,62**	**3537483,81**	**332,51**
DLECSA2	10042,58	2,79	328972070,26	4129,99	**333037561,88**	**4061361,63**	**333,04**
DLECSA3	9996,08	2,78	329021073,33	4007,86	**332519024,34**	**3493943,15**	**332,52**
ECI	15203,62	4,22	327069693,45	9726,84	**332363567,56**	**5284147,27**	**332,36**
EWQ	7364,18	2,05	329051439,88	241,38	**332719803,52**	**3668122,26**	**332,72**
ESUFF	8466,69	2,35	342190250,73	2387,03	**345861129,33**	**3668491,57**	**345,86**
EXSUFF	18455,42	5,13	329229186,65	13872,65	**332736775,87**	**3493716,56**	**332,74**
EDFPLTF	7965,58	2,21	329253327,41	1743,61	**332880072,34**	**3625001,33**	**332,88**

Plataforma 1 – 144 tarefas

0%							
Algoritmos	**Tempo (seg)**	**Tempo (horas)**	**Consumo (watts)**	**Cons. Ocioso (watts)**	**CONSUMO TOTAL**	**Rede (watts)**	**C. T. Mega Watts**
WQ	8341,50	2,32	328379395,42	237,93	**332746569,09**	**4366935,74**	**332,75**
WQR	8432,33	2,34	353370089,50	2101,99	**357739128,14**	**4366936,65**	**357,74**
Suff	9059,92	2,52	336371554,96	2871,57	**340742571,22**	**4368144,69**	**340,74**
XSuff	9062,26	2,52	329304000,41	3144,20	**331492139,08**	**2184994,47**	**331,49**
DFPLTF	8225,37	2,28	328379395,42	1816,79	**332749356,38**	**4368144,17**	**332,75**
LECSA	28445,34	7,90	326827054,02	666,69	**326827743,18**	**22,47**	**326,83**
LECSA2	28445,34	7,90	326827054,02	666,69	**326827743,18**	**22,47**	**326,83**
LECSA3	26227,95	7,29	326975476,84	631,43	**326976137,52**	**29,25**	**326,98**
DLECSA	28445,34	7,90	326827054,02	666,69	**326827743,18**	**22,47**	**326,83**
DLECSA2	28445,34	7,90	326827054,02	666,69	**326827743,18**	**22,47**	**326,83**
DLECSA3	26227,95	7,29	326975476,84	631,43	**326976137,52**	**29,25**	**326,98**
ECI	13513,42	3,75	327043512,57	7752,35	**331419840,19**	**4368575,27**	**331,42**
EWQ	8238,50	2,29	328379395,42	221,11	**332746555,32**	**4366938,79**	**332,75**
ESUFF	8189,94	2,27	334052445,50	1771,76	**338422928,46**	**4368711,20**	**338,42**
EXSUFF	9062,26	2,52	329304000,41	3144,20	**331492139,08**	**2184994,47**	**331,49**
EDFPLTF	8189,94	2,27	328373211,13	1756,16	**330560305,41**	**2185338,12**	**330,56**

25%							
Algoritmos	**Tempo (seg)**	**Tempo (horas)**	**Consumo (watts)**	**Cons. Ocioso (watts)**	**CONSUMO TOTAL**	**Rede (watts)**	**C. T. Mega Watts**
WQ	9009,80	2,50	328894488,87	237,60	**333523669,13**	**4628942,65**	**333,52**
WQR	9101,74	2,53	339516766,94	3073,94	**344148784,06**	**4628943,18**	**344,15**
Suff	9368,63	2,60	328503593,34	3273,30	**330964567,19**	**2457700,56**	**330,96**
XSuff	10112,65	2,81	329253363,57	4317,02	**331617377,31**	**2359696,72**	**331,62**
DFPLTF	8649,28	2,40	328821463,35	2092,63	**333191840,91**	**4368284,93**	**333,19**
LECSA	31983,08	8,88	326827054,02	1832,22	**326828908,78**	**22,54**	**326,83**
LECSA2	31983,08	8,88	326827054,02	1832,22	**326828908,78**	**22,54**	**326,83**
LECSA3	29489,91	8,19	326975476,84	1738,47	**326977244,71**	**29,40**	**326,98**
DLECSA	37465,99	10,41	326827054,02	3111,81	**326830188,79**	**22,97**	**326,83**
DLECSA2	37465,99	10,41	326827054,02	3111,81	**326830188,79**	**22,97**	**326,83**
DLECSA3	34558,09	9,60	326975476,84	2958,59	**326978464,74**	**29,32**	**326,98**
ECI	13615,27	3,78	327052747,65	7905,07	**331789548,70**	**4728895,97**	**331,79**
EWQ	9186,63	2,55	328740795,18	218,00	**333369959,37**	**4628946,19**	**333,37**
ESUFF	9186,07	2,55	334113809,71	3010,36	**338300011,54**	**4183191,47**	**338,30**
EXSUFF	10112,65	2,81	329253363,57	4317,02	**331617377,31**	**2359696,72**	**331,62**
EDFPLTF	8861,40	2,46	329708001,55	2999,14	**333894334,76**	**4183334,07**	**333,89**

50%							
Algoritmos	**Tempo (seg)**	**Tempo (horas)**	**Consumo (watts)**	**Cons. Ocioso (watts)**	**CONSUMO TOTAL**	**Rede (watts)**	**C. T. Mega Watts**
WQ	9295,30	2,58	329127170,95	237,41	**333057668,09**	**3930259,73**	**333,06**
WQR	9387,42	2,61	338373658,16	3483,88	**342307402,17**	**3930260,13**	**342,31**
Suff	9422,78	2,62	333370190,73	3330,16	**336104099,87**	**2730578,98**	**336,10**
XSuff	11226,05	3,12	329306969,04	5751,49	**331497715,83**	**2184995,29**	**331,50**
DFPLTF	8601,24	2,39	329443392,07	2416,43	**333814239,39**	**4368430,88**	**333,81**
LECSA	20340,65	5,65	326871549,94	5267,64	**326876941,19**	**123,61**	**326,88**
LECSA2	18756,70	5,21	326904275,11	4286,12	**326908684,79**	**123,55**	**326,91**
LECSA3	20342,71	5,65	326918560,81	5547,03	**329871790,22**	**2947682,38**	**329,87**
DLECSA	20136,91	5,59	326883073,33	5162,74	**326888355,90**	**119,83**	**326,89**
DLECSA2	17906,97	4,97	326891473,64	3612,75	**326895203,49**	**117,09**	**326,90**
DLECSA3	17531,19	4,87	326937546,56	3166,62	**330379648,95**	**3438935,78**	**330,38**
ECI	13676,10	3,80	327051294,18	7978,61	**332301238,27**	**5241965,48**	**332,30**
EWQ	9250,58	2,57	328998767,27	217,77	**331619222,77**	**2620237,74**	**331,62**
ESUFF	10387,17	2,89	336737733,31	4512,07	**339364300,18**	**2622054,80**	**339,36**
EXSUFF	11226,05	3,12	329306969,04	5751,49	**331497715,83**	**2184995,29**	**331,50**
EDFPLTF	8614,68	2,39	329525070,16	2696,67	**333678566,37**	**4150799,55**	**333,68**

75%							
Algoritmos	**Tempo (seg)**	**Tempo (horas)**	**Consumo (watts)**	**Cons. Ocioso (watts)**	**CONSUMO TOTAL**	**Rede (watts)**	**C. T. Mega Watts**
WQ	10511,75	2,92	329137401,29	237,48	**332139973,54**	**3002334,78**	**332,14**
WQR	10603,43	2,95	339624212,94	4956,36	**342631504,59**	**3002335,28**	**342,63**
Suff	10238,63	2,84	344294192,52	4064,11	**347301806,79**	**3003550,16**	**347,30**
XSuff	12339,22	3,43	329327883,48	6821,31	**332338531,73**	**3003826,95**	**332,34**
DFPLTF	8630,23	2,40	329402954,91	2644,44	**332114788,15**	**2709188,80**	**332,11**
LECSA	16796,84	4,67	327318162,82	8046,42	**329358323,05**	**2032113,81**	**329,36**
LECSA2	15489,91	4,30	327318163,82	8047,42	**330577368,70**	**3251157,47**	**330,58**
LECSA3	16798,39	4,67	327318162,82	8046,47	**330577380,17**	**3251170,88**	**330,58**
DLECSA	13787,24	3,83	327570433,47	3929,49	**331109959,78**	**3535596,82**	**331,11**
DLECSA2	13500,98	3,75	327410317,97	4520,49	**330665978,21**	**3251139,75**	**330,67**
DLECSA3	15288,78	4,25	327546159,08	4032,04	**331800978,35**	**4250787,23**	**331,80**
ECI	18016,42	5,00	327083300,04	12971,44	**332065402,93**	**4969131,45**	**332,07**
EWQ	10467,54	2,91	329062948,55	217,78	**332065503,21**	**3002336,88**	**332,07**
ESUFF	10270,88	2,85	341096370,77	4112,78	**344104718,98**	**3004235,43**	**344,10**
EXSUFF	12339,22	3,43	329327883,48	6821,31	**332338531,73**	**3003826,95**	**332,34**
EDFPLTF	8597,61	2,39	329455755,82	2491,09	**333227143,03**	**3768896,12**	**333,23**

100%							
Algoritmos	**Tempo (seg)**	**Tempo (horas)**	**Consumo (watts)**	**Cons. Ocioso (watts)**	**CONSUMO TOTAL**	**Rede (watts)**	**C. T. Mega Watts**
WQ	10500,87	2,92	329245039,84	237,40	**332302189,45**	**3056912,20**	**332,30**
WQR	10593,29	2,94	339064147,20	4981,87	**342126041,70**	**3056912,64**	**342,13**
Suff	9541,78	2,65	343679514,58	3474,05	**348487753,89**	**4804765,26**	**348,49**
XSuff	13451,89	3,74	329258440,27	8413,02	**332325236,48**	**3058383,19**	**332,33**
DFPLTF	9013,91	2,50	329280601,77	2868,70	**332342148,04**	**3058677,57**	**332,34**
LECSA	12241,91	3,40	327779279,85	6426,88	**329316661,71**	**1530954,98**	**329,32**
LECSA2	11290,29	3,14	327817828,56	5396,60	**332191776,59**	**4368551,43**	**332,19**
LECSA3	12241,91	3,40	327779279,85	6425,55	**332155970,83**	**4370265,42**	**332,16**
DLECSA	12241,36	3,40	328296109,13	6639,26	**331579912,02**	**3277163,63**	**331,58**
DLECSA2	10902,13	3,03	329067334,41	4719,37	**333003547,83**	**3931494,04**	**333,00**
DLECSA3	10506,63	2,92	329020846,34	3937,56	**332302592,19**	**3277808,29**	**332,30**
ECI	17802,62	4,95	327054644,86	12757,00	**333619502,09**	**6552100,23**	**333,62**
EWQ	10512,60	2,92	329170721,05	218,00	**332446189,01**	**3275249,96**	**332,45**
ESUFF	11243,81	3,12	340677009,96	4491,38	**343958670,73**	**3277169,39**	**343,96**
EXSUFF	13451,89	3,74	329258440,27	8413,02	**332325236,48**	**3058383,19**	**332,33**
EDFPLTF	9011,85	2,50	329302763,27	2622,30	**332583249,41**	**3277863,84**	**332,58**

Plataforma 1 – 29 tarefas

0%

Algoritmos	Tempo (seg)	Tempo (horas)	Consumo (watts)	Cons. Ocioso (watts)	CONSUMO TOTAL	Rede (watts)	C. T. Mega Watts
WQ	22419,65	6,23	331622079,16	233,64	**331622759,89**	**447,09**	**331,62**
WQR	22511,60	6,25	389599815,71	19299,62	**389619574,86**	**459,54**	**389,62**
Suff	18749,27	5,21	329246139,87	13727,89	**340181842,44**	**10921974,68**	**340,18**
XSuff	22355,77	6,21	331622079,16	19110,39	**331647537,07**	**6347,52**	**331,65**
DFPLTF	18749,27	5,21	329246139,87	13727,89	**340181842,44**	**10921974,68**	**340,18**
LECSA	20336,74	5,65	79437131,18	53,42	**79437194,40**	**9,79**	**79,44**
LECSA2	20336,74	5,65	79437131,18	53,42	**79437194,40**	**9,79**	**79,44**
LECSA3	18751,53	5,21	79473206,18	50,19	**79473269,06**	**12,69**	**79,47**
DLECSA	101310,57	28,14	329096686,34	1871,82	**329098568,03**	**9,88**	**329,10**
DLECSA2	101310,57	28,14	329096686,34	1871,82	**329098568,03**	**9,88**	**329,10**
DLECSA3	93385,44	25,94	329246139,87	1777,64	**329247930,32**	**12,80**	**329,25**
ECI	20334,99	5,65	329106993,48	15489,69	**329130299,14**	**7815,97**	**329,13**
EWQ	20389,92	5,66	329106993,48	214,56	**329107619,28**	**411,24**	**329,11**
ESUFF	20334,99	5,65	329106993,48	15489,69	**329130299,14**	**7815,97**	**329,13**
EXSUFF	22355,77	6,21	331622079,16	19110,39	**331647537,07**	**6347,52**	**331,65**
EDFPLTF	20334,99	5,65	329106993,48	15489,69	**329130299,14**	**7815,97**	**329,13**

25%

Algoritmos	Tempo (seg)	Tempo (horas)	Consumo (watts)	Cons. Ocioso (watts)	CONSUMO TOTAL	Rede (watts)	C. T. Mega Watts
WQ	25168,65	6,99	331609745,95	233,60	**331610425,44**	**445,89**	**331,61**
WQR	25260,72	7,02	344654736,67	22606,58	**344677802,30**	**459,04**	**344,68**
Suff	21081,36	5,86	329246139,87	16479,41	**341549361,39**	**12286742,10**	**341,55**
XSuff	25138,19	6,98	331609745,95	22457,72	**331638738,66**	**6535,00**	**331,64**
DFPLTF	21081,36	5,86	329246139,87	16479,41	**338820377,69**	**9557758,41**	**338,82**
LECSA	22867,11	6,35	89366772,58	53,42	**89366837,03**	**11,02**	**89,37**
LECSA2	22867,11	6,35	89366772,58	53,42	**89366837,03**	**11,02**	**89,37**
LECSA3	21083,62	5,86	89407356,95	50,19	**89407421,42**	**14,28**	**89,41**
DLECSA	108988,56	30,27	329096686,34	2600,60	**329099296,64**	**9,70**	**329,10**
DLECSA2	108988,56	30,27	329096686,34	2600,60	**329099296,64**	**9,70**	**329,10**
DLECSA3	100463,81	27,91	329246139,87	2470,00	**329248622,42**	**12,56**	**329,25**
ECI	22865,36	6,35	329106993,48	18513,02	**329133647,46**	**8140,96**	**329,13**
EWQ	22903,92	6,36	329106993,48	213,82	**329107622,61**	**415,31**	**329,11**
ESUFF	22865,36	6,35	329106993,48	18513,02	**329133647,46**	**8140,96**	**329,13**
EXSUFF	25138,19	6,98	331609745,95	22457,72	**331638738,66**	**6535,00**	**331,64**
EDFPLTF	22865,36	6,35	329105730,85	18509,29	**329132700,31**	**8460,17**	**329,13**

50%							
Algoritmos	**Tempo (seg)**	**Tempo (horas)**	**Consumo (watts)**	**Cons. Ocioso (watts)**	**CONSUMO TOTAL**	**Rede (watts)**	**C. T. Mega Watts**
WQ	27983,65	7,77	331656146,12	233,61	**331656825,51**	**445,78**	**331,66**
WQR	28075,42	7,80	353397797,33	26018,16	**353424275,95**	**460,46**	**353,42**
Suff	23413,45	6,50	329246139,87	19230,94	**342370581,72**	**13105210,91**	**342,37**
XSuff	27920,92	7,76	331656146,12	25830,77	**331688338,67**	**6361,78**	**331,69**
DFPLTF	23413,45	6,50	329246139,87	19230,94	**337458901,12**	**8193530,31**	**337,46**
LECSA	25397,31	7,05	187925556,06	338,57	**187925944,75**	**50,12**	**187,93**
LECSA2	25397,31	7,05	187925556,06	338,57	**187925944,75**	**50,12**	**187,93**
LECSA3	23415,71	6,50	188010899,18	320,24	**188011283,47**	**64,04**	**188,01**
DLECSA	69932,66	19,43	329096686,34	5902,32	**329102631,14**	**42,48**	**329,10**
DLECSA2	69932,66	19,43	329096686,34	5902,32	**329102631,14**	**42,48**	**329,10**
DLECSA3	64463,79	17,91	329246139,87	5604,61	**329251795,78**	**51,30**	**329,25**
ECI	25395,72	7,05	329108024,19	21539,40	**329137764,21**	**8200,61**	**329,14**
EWQ	25444,92	7,07	329108024,19	213,80	**329108651,86**	**413,87**	**329,11**
ESUFF	25395,72	7,05	329108024,19	21539,41	**329137764,21**	**8200,61**	**329,14**
EXSUFF	27920,92	7,76	331656146,12	25830,77	**331688338,67**	**6361,78**	**331,69**
EDFPLTF	25395,72	7,05	329104416,69	21528,74	**329135041,83**	**9096,40**	**329,14**

75%							
Algoritmos	**Tempo (seg)**	**Tempo (horas)**	**Consumo (watts)**	**Cons. Ocioso (watts)**	**CONSUMO TOTAL**	**Rede (watts)**	**C. T. Mega Watts**
WQ	30758,65	8,54	331794826,54	233,60	**331795507,78**	**447,64**	**331,80**
WQR	30849,72	8,57	347738704,09	29403,78	**347768571,57**	**463,70**	**347,77**
Suff	25745,54	7,15	329246139,87	21982,46	**343683428,44**	**14415306,11**	**343,68**
XSuff	30703,74	8,53	331794826,54	29226,64	**331830393,14**	**6339,97**	**331,83**
DFPLTF	25745,54	7,15	329246139,87	21982,46	**336097439,99**	**6829317,66**	**336,10**
LECSA	27927,67	7,76	272355878,35	1533,25	**272357527,80**	**116,20**	**272,36**
LECSA2	27927,67	7,76	272355878,35	1533,25	**272357527,80**	**116,20**	**272,36**
LECSA3	25747,80	7,15	272479564,03	1453,94	**272481164,36**	**146,39**	**272,48**
DLECSA	55762,37	15,49	329096686,34	8540,93	**329105331,80**	**104,53**	**329,11**
DLECSA2	55762,37	15,49	329096686,34	8540,93	**329105331,80**	**104,53**	**329,11**
DLECSA3	51401,67	14,28	329246139,87	8109,16	**329254378,38**	**129,35**	**329,25**
ECI	27926,08	7,76	329110085,62	24568,84	**329142915,33**	**8260,86**	**329,14**
EWQ	27966,92	7,77	329110085,62	213,82	**329110713,01**	**413,57**	**329,11**
ESUFF	27926,08	7,76	329110085,62	24568,84	**329142915,33**	**8260,86**	**329,14**
EXSUFF	30703,74	8,53	331794826,54	29226,64	**331830393,14**	**6339,97**	**331,83**
EDFPLTF	27926,08	7,76	329103128,30	24548,26	**329127880,50**	**203,94**	**329,13**

100%							
Algoritmos	**Tempo (seg)**	**Tempo (horas)**	**Consumo (watts)**	**Cons. Ocioso (watts)**	**CONSUMO TOTAL**	**Rede (watts)**	**C. T. Mega Watts**
WQ	33490,65	9,30	331554150,80	233,58	**331554841,05**	**456,66**	**331,55**
WQR	33581,58	9,33	340250811,28	32626,08	**340283911,47**	**474,10**	**340,28**
Suff	28077,63	7,80	329246139,87	24733,98	**345650971,03**	**16380097,18**	**345,65**
XSuff	33486,37	9,30	331554150,80	32510,15	**331593188,96**	**6528,01**	**331,59**
DFPLTF	28077,63	7,80	329246139,87	24733,98	**334735862,87**	**5464989,01**	**334,74**
LECSA	30458,04	8,46	329101839,91	4494,23	**329106542,37**	**208,23**	**329,11**
LECSA2	30457,77	8,46	329112147,05	4451,78	**329116807,70**	**208,87**	**329,12**
LECSA3	28079,89	7,80	329246139,87	4229,16	**331009441,85**	**1759072,82**	**331,01**
DLECSA	30458,72	8,46	329107508,83	4511,08	**329112224,87**	**204,95**	**329,11**
DLECSA2	30461,57	8,46	329106993,48	3643,98	**329110835,08**	**197,62**	**329,11**
DLECSA3	28079,47	7,80	329246139,87	4228,99	**334527063,20**	**5276694,34**	**334,53**
ECI	30456,45	8,46	329107508,83	27584,55	**329143618,54**	**8525,16**	**329,14**
EWQ	30488,92	8,47	329107508,83	213,86	**329108138,20**	**415,50**	**329,11**
ESUFF	30456,45	8,46	329107508,83	27584,55	**329143618,55**	**8525,16**	**329,14**
EXSUFF	33486,37	9,30	331554150,80	32510,15	**331593188,96**	**6528,01**	**331,59**
EDFPLTF	30456,45	8,46	329101839,91	27567,78	**329139640,37**	**10232,68**	**329,14**

Plataforma 2 – 720 tarefas

0%							
Algoritmos	**Tempo (seg)**	**Tempo (horas)**	**Consumo (watts)**	**Cons. Ocioso (watts)**	**CONSUMO TOTAL**	**Rede (watts)**	**C. T. Mega Watts**
WQ	13314,54	3,70	393974614,68	278,87	**393974893,55**	**3930125,93**	**393,97**
WQR	13405,19	3,72	409779212,38	8057,04	**409787269,42**	**3930126,00**	**409,79**
Suff	7798,15	2,17	377055383,43	2086,99	**377057470,42**	**4367028,93**	**377,06**
XSuff	144880,31	40,24	582031351,48	155712,78	**582187064,26**	**3057039,46**	**582,19**
DFPLTF	11937,83	3,32	396164580,76	6908,09	**396171488,86**	**3930351,73**	**396,17**
LECSA	26874,34	7,47	326846843,73	241,18	**326847084,91**	**6,47**	**326,85**
LECSA2	26874,34	7,47	326847462,16	208,25	**326847670,41**	**6,37**	**326,85**
LECSA3	23214,90	6,45	336291452,59	141,20	**336291593,79**	**7,12**	**336,29**
DLECSA	26926,41	7,48	326848080,58	261,16	**326848341,74**	**6,47**	**326,85**
DLECSA2	26920,90	7,48	326848080,58	259,64	**326848340,22**	**6,44**	**326,85**
DLECSA3	23214,90	6,45	336291452,59	141,20	**336291593,79**	**7,12**	**336,29**
ECI	13245,14	3,68	355030755,94	8565,34	**355039321,28**	**3930551,38**	**355,04**
EWQ	13342,86	3,71	391554977,92	290,66	**391555268,58**	**3930126,98**	**391,56**
ESUFF	17683,08	4,91	376477478,87	14113,12	**376491591,99**	**4367231,54**	**376,49**
EXSUFF	133803,15	37,17	582031351,48	148185,94	**582179537,42**	**3493712,70**	**582,18**
EDFPLTF	43807,88	12,17	425042481,84	41083,96	**425083565,80**	**3930555,09**	**425,08**

25%							
Algoritmos	**Tempo (seg)**	**Tempo (horas)**	**Consumo (watts)**	**Cons. Ocioso (watts)**	**CONSUMO TOTAL**	**Rede (watts)**	**C. T. Mega Watts**
WQ	14926,61	4,15	394860840,99	275,79	**398806526,26**	**3945409,48**	**398,81**
WQR	15018,76	4,17	407568120,77	9947,57	**411523477,89**	**3945409,56**	**411,52**
Suff	10386,62	2,89	438233978,44	5562,34	**440741972,10**	**2502431,32**	**440,74**
XSuff	164916,34	45,81	582531157,22	186333,70	**585907717,00**	**3190226,08**	**585,91**
DFPLTF	11575,17	3,22	394084880,02	6448,02	**398098129,25**	**4006801,21**	**398,10**
LECSA	30200,79	8,39	326844431,86	1335,75	**326845774,00**	**6,39**	**326,85**
LECSA2	30200,79	8,39	326850013,17	1312,22	**326851331,60**	**6,21**	**326,85**
LECSA3	26097,50	7,25	336291452,59	1137,55	**336292597,07**	**6,93**	**336,29**
DLECSA	33559,03	9,32	326848454,73	2459,32	**326850920,21**	**6,16**	**326,85**
DLECSA2	34029,80	9,45	326850771,78	2621,71	**326853399,58**	**6,09**	**326,85**
DLECSA3	30006,09	8,34	336291452,59	2481,66	**336293940,99**	**6,74**	**336,29**
ECI	13976,59	3,88	356377431,93	9483,32	**360428825,72**	**4041910,46**	**360,43**
EWQ	14976,13	4,16	393920623,97	257,18	**398095545,95**	**4174664,80**	**398,10**
ESUFF	15369,83	4,27	376811811,32	9295,78	**381225461,81**	**4404354,71**	**381,23**
EXSUFF	139065,58	38,63	582531157,22	154427,80	**586257900,92**	**3572315,90**	**586,26**
EDFPLTF	51913,81	14,42	444489125,04	50651,40	**448114498,22**	**3574721,78**	**448,11**

50%							
Algoritmos	**Tempo (seg)**	**Tempo (horas)**	**Consumo (watts)**	**Cons. Ocioso (watts)**	**CONSUMO TOTAL**	**Rede (watts)**	**C. T. Mega Watts**
WQ	16545,65	4,60	398458150,90	275,84	**402432220,02**	**3973793,28**	**402,43**
WQR	16636,60	4,62	408432826,24	12011,29	**412418630,90**	**3973793,38**	**412,42**
Suff	9909,04	2,75	429976742,60	4779,95	**433431532,06**	**3450009,51**	**433,43**
XSuff	178578,28	49,61	588764437,08	198965,86	**592107773,10**	**3144370,16**	**592,11**
DFPLTF	11612,43	3,23	394015057,81	6496,41	**398126638,80**	**4105084,57**	**398,13**
LECSA	16316,51	4,53	327366114,08	2440,47	**330163300,20**	**2794745,66**	**330,16**
LECSA2	15039,94	4,18	327386377,91	1654,19	**330881457,18**	**3493425,08**	**330,88**
LECSA3	15039,94	4,18	331618585,03	2440,09	**335114451,21**	**3493426,08**	**335,11**
DLECSA	17565,23	4,88	327444771,79	3323,27	**330985185,15**	**3537090,09**	**330,99**
DLECSA2	17624,44	4,90	327390460,12	3354,65	**331203826,50**	**3810011,73**	**331,20**
DLECSA3	15867,62	4,41	332458963,68	2489,61	**335889376,33**	**3427923,04**	**335,89**
ECI	12918,60	3,59	354393073,44	8147,62	**358506445,02**	**4105223,95**	**358,51**
EWQ	16608,49	4,61	396836682,20	256,28	**400963569,17**	**4126630,69**	**400,96**
ESUFF	24865,94	6,91	390040511,91	20145,24	**394689896,84**	**4629239,68**	**394,69**
EXSUFF	149120,87	41,42	588764437,08	166638,70	**592446616,65**	**3515540,87**	**592,45**
EDFPLTF	56542,67	15,71	452655002,91	56214,57	**456707387,22**	**3996169,74**	**456,71**

75%							
Algoritmos	**Tempo (seg)**	**Tempo (horas)**	**Consumo (watts)**	**Cons. Ocioso (watts)**	**CONSUMO TOTAL**	**Rede (watts)**	**C. T. Mega Watts**
WQ	18178,05	5,05	395310829,89	275,81	**399459567,44**	**4148461,75**	**399,46**
WQR	18268,77	5,07	401169044,33	13856,56	**405331362,71**	**4148461,82**	**405,33**
Suff	16260,31	4,52	433796258,00	11244,62	**437672343,50**	**3864840,88**	**437,67**
XSuff	214162,40	59,49	586764787,69	247523,87	**590909935,80**	**3897624,25**	**590,91**
DFPLTF	11189,07	3,11	390340029,18	5962,46	**394320105,20**	**3974113,57**	**394,32**
LECSA	12355,33	3,43	329201651,49	3848,54	**332586415,62**	**3380915,59**	**332,59**
LECSA2	10678,10	2,97	330599660,58	2273,25	**333999106,67**	**3397172,85**	**334,00**
LECSA3	12359,88	3,43	329501343,04	4371,51	**332886627,49**	**3380912,94**	**332,89**
DLECSA	13396,42	3,72	330275596,11	4979,66	**333507069,08**	**3226493,31**	**333,51**
DLECSA2	12196,31	3,39	330300161,46	3391,33	**333603191,08**	**3299638,29**	**333,60**
DLECSA3	11859,70	3,29	330626169,03	3398,58	**333888566,68**	**3258999,07**	**333,89**
ECI	15009,61	4,17	358370283,27	10778,68	**362147858,19**	**3766796,23**	**362,15**
EWQ	18216,19	5,06	393129520,45	255,94	**397332823,74**	**4203047,36**	**397,33**
ESUFF	20316,68	5,64	389126485,39	15561,48	**393531110,84**	**4389063,98**	**393,53**
EXSUFF	177310,22	49,25	586764787,69	198207,16	**590554946,75**	**3591951,90**	**590,55**
EDFPLTF	66101,78	18,36	478189036,42	66465,43	**481869637,82**	**3614135,98**	**481,87**

100%							
Algoritmos	**Tempo (seg)**	**Tempo (horas)**	**Consumo (watts)**	**Cons. Ocioso (watts)**	**CONSUMO TOTAL**	**Rede (watts)**	**C. T. Mega Watts**
WQ	19803,15	5,50	394590242,13	276,26	**398476976,49**	**3886458,10**	**398,48**
WQR	15487,09	4,30	387636930,01	11262,50	**391534650,72**	**3886458,21**	**391,53**
Suff	13072,48	3,63	430398046,22	7383,71	**431584729,97**	**1179300,04**	**431,58**
XSuff	229554,66	63,77	586345685,40	265146,85	**590366538,91**	**3755706,67**	**590,37**
DFPLTF	16897,77	4,69	393335616,47	11889,21	**397277983,55**	**3930477,87**	**397,28**
LECSA	52353,59	14,54	455388931,10	45661,94	**458928660,52**	**3494067,48**	**458,93**
LECSA2	52353,59	14,54	457149198,45	45538,88	**460688714,20**	**3493976,87**	**460,69**
LECSA3	52353,59	14,54	455388931,10	45471,57	**458928314,11**	**3493911,44**	**458,93**
DLECSA	87872,70	24,41	492245061,78	76115,80	**496513752,62**	**4192575,04**	**496,51**
DLECSA2	35451,37	9,85	420642756,29	13503,20	**424761433,28**	**4105173,79**	**424,76**
DLECSA3	45981,08	12,77	457589123,96	35717,88	**461293314,14**	**3668472,30**	**461,29**
ECI	15859,76	4,41	359825770,62	11848,49	**364030189,23**	**4192570,12**	**364,03**
EWQ	19874,56	5,52	394110402,58	256,63	**398084453,99**	**3973794,79**	**398,08**
ESUFF	23261,70	6,46	394004235,39	18748,12	**398433899,58**	**4410916,07**	**398,43**
EXSUFF	187921,38	52,20	586345685,40	216435,78	**590317826,35**	**3755705,17**	**590,32**
EDFPLTF	66383,44	18,44	477962010,73	66893,51	**481610346,29**	**3581442,06**	**481,61**

Plataforma 2 – 144 tarefas

0%

Algoritmos	Tempo (seg)	Tempo (horas)	Consumo (watts)	Cons. Ocioso (watts)	CONSUMO TOTAL	Rede (watts)	C. T. Mega Watts
WQ	65560,71	18,21	488061265,39	250,99	**492428452,32**	**4366935,94**	**492,43**
WQR	48819,44	13,56	450675058,66	53219,60	**455095214,64**	**4366936,38**	**455,10**
Suff	9064,85	2,52	375236117,93	3322,42	**379607508,42**	**4368068,08**	**379,61**
XSuff	130712,65	36,31	620801760,56	134104,72	**623120820,69**	**2184955,42**	**623,12**
DFPLTF	8227,93	2,29	367079427,66	2074,35	**371449569,27**	**4368067,26**	**371,45**
LECSA	28445,58	7,90	326845606,87	759,35	**326846390,70**	**24,47**	**326,85**
LECSA2	28445,58	7,90	326848699,01	666,16	**326849389,10**	**23,93**	**326,85**
LECSA3	24564,01	6,82	336291452,59	607,51	**336292086,04**	**25,93**	**336,29**
DLECSA	28462,01	7,91	326848699,01	773,78	**326849497,27**	**24,47**	**326,85**
DLECSA2	28462,31	7,91	326848699,01	773,87	**326849497,21**	**24,32**	**326,85**
DLECSA3	24564,01	6,82	336291452,59	607,51	**336292086,04**	**25,93**	**336,29**
ECI	13236,00	3,68	354687315,43	8562,42	**359064972,33**	**4369094,48**	**359,06**
EWQ	65589,71	18,22	488061265,39	233,12	**492428440,14**	**4366941,63**	**492,43**
ESUFF	9035,57	2,51	336641351,46	3316,56	**341013758,12**	**4369090,09**	**341,01**
EXSUFF	130712,65	36,31	620801760,56	134104,93	**623120818,68**	**2184953,18**	**623,12**
EDFPLTF	9035,57	2,51	330622397,87	3255,18	**334994742,58**	**4369089,54**	**334,99**

25%

Algoritmos	Tempo (seg)	Tempo (horas)	Consumo (watts)	Cons. Ocioso (watts)	CONSUMO TOTAL	Rede (watts)	C. T. Mega Watts
WQ	73690,85	20,47	492612148,83	251,20	**494970635,23**	**2358235,19**	**494,97**
WQR	48826,87	13,56	452115639,09	53227,96	**454527126,36**	**2358259,30**	**454,53**
Suff	9260,52	2,57	366736774,78	3365,26	**370835310,45**	**4095170,41**	**370,84**
XSuff	147038,82	40,84	624470822,84	150728,04	**626981209,52**	**2359658,64**	**626,98**
DFPLTF	8206,10	2,28	367125820,47	1983,35	**371397754,16**	**4269950,33**	**371,40**
LECSA	31980,04	8,88	326843287,76	1921,90	**326845234,46**	**24,80**	**326,85**
LECSA2	31276,70	8,69	326851353,10	1630,54	**326853007,57**	**23,93**	**326,85**
LECSA3	27621,31	7,67	336291452,59	1664,25	**336293142,81**	**25,97**	**336,29**
DLECSA	32588,90	9,05	326847132,33	2128,57	**326849285,77**	**24,87**	**326,85**
DLECSA2	34203,64	9,50	326850626,45	2611,13	**326853262,37**	**24,79**	**326,85**
DLECSA3	32364,69	8,99	336291452,59	2827,14	**336294306,10**	**26,36**	**336,29**
ECI	13758,47	3,82	355627072,82	9211,76	**359918119,64**	**4281835,06**	**359,92**
EWQ	73747,37	20,49	489374800,04	231,48	**491831522,84**	**2456491,33**	**491,83**
ESUFF	10876,10	3,02	345905111,48	5232,08	**350334093,02**	**4423749,46**	**350,33**
EXSUFF	147038,82	40,84	624470822,84	152656,12	**626983134,00**	**2359655,03**	**626,98**
EDFPLTF	8946,52	2,49	331167902,52	3190,97	**335627782,70**	**4456689,21**	**335,63**

50%							
Algoritmos	**Tempo (seg)**	**Tempo (horas)**	**Consumo (watts)**	**Cons. Ocioso (watts)**	**CONSUMO TOTAL**	**Rede (watts)**	**C. T. Mega Watts**
WQ	81799,99	22,72	488850185,38	250,97	**493763214,34**	**4912777,99**	**493,76**
WQR	52790,55	14,66	453492588,16	58455,44	**458463822,02**	**4912778,42**	**458,46**
Suff	10353,04	2,88	364694231,45	4338,87	**366337416,26**	**1638845,94**	**366,34**
XSuff	163436,16	45,40	636695085,87	171119,93	**639051159,24**	**2184953,44**	**639,05**
DFPLTF	8608,50	2,39	366216235,05	2628,38	**368665839,62**	**2446976,19**	**368,67**
LECSA	20341,20	5,65	327331181,46	5082,57	**329847260,51**	**2510996,48**	**329,85**
LECSA2	18754,46	5,21	327361937,96	4120,71	**330641235,02**	**3275176,35**	**330,64**
LECSA3	18754,74	5,21	331250474,60	5032,59	**334530687,96**	**3275180,77**	**334,53**
DLECSA	21005,62	5,83	327449276,67	5561,17	**330948345,13**	**3493507,28**	**330,95**
DLECSA2	17495,63	4,86	327351146,39	3225,95	**331339140,87**	**3984768,53**	**331,34**
DLECSA3	17983,98	5,00	332338632,32	4383,59	**334635670,51**	**2292654,60**	**334,64**
ECI	11647,00	3,24	378143335,21	6495,11	**383173959,85**	**5024129,54**	**383,17**
EWQ	78645,26	21,85	490372772,76	231,10	**495176616,87**	**4803613,01**	**495,18**
ESUFF	12321,09	3,42	349876609,88	5402,21	**355015313,27**	**5133301,19**	**355,02**
EXSUFF	160100,48	44,47	636695085,87	165262,71	**639045301,65**	**2184953,07**	**639,05**
EDFPLTF	9510,44	2,64	331015308,38	3862,41	**335388795,62**	**4369624,84**	**335,39**

75%							
Algoritmos	**Tempo (seg)**	**Tempo (horas)**	**Consumo (watts)**	**Cons. Ocioso (watts)**	**CONSUMO TOTAL**	**Rede (watts)**	**C. T. Mega Watts**
WQ	90031,18	25,01	520888786,94	251,06	**523891373,03**	**3002335,03**	**523,89**
WQR	82337,26	22,87	602954115,20	94874,43	**606051324,95**	**3002335,32**	**606,05**
Suff	11125,34	3,09	366084176,27	5505,68	**370163035,25**	**4073353,30**	**370,16**
XSuff	168262,65	46,74	639593944,17	174713,32	**642772437,25**	**3003779,76**	**642,77**
DFPLTF	8585,32	2,38	364788667,93	2479,31	**369377941,62**	**4586794,38**	**369,38**
LECSA	16797,25	4,67	329114980,86	8123,63	**332179227,01**	**3056122,52**	**332,18**
LECSA2	14509,90	4,03	331100573,44	6017,16	**334162721,50**	**3056130,90**	**334,16**
LECSA3	14509,90	4,03	331100573,44	6017,16	**334162721,50**	**3056130,90**	**334,16**
DLECSA	16090,56	4,47	329781641,27	7343,06	**331821096,39**	**2032112,06**	**331,82**
DLECSA2	14047,95	3,90	330510806,37	4746,57	**333766719,16**	**3251166,22**	**333,77**
DLECSA3	14227,43	3,95	329715671,81	4116,75	**333564205,34**	**3844416,78**	**333,56**
ECI	13236,00	3,68	354671060,65	8540,83	**360391613,51**	**5712012,03**	**360,39**
EWQ	90061,25	25,02	496188571,21	231,23	**498918223,47**	**2729421,03**	**498,92**
ESUFF	12234,02	3,40	356370753,81	6665,09	**360746660,26**	**4369241,35**	**360,75**
EXSUFF	168262,65	46,74	639593944,17	176279,51	**642774009,79**	**3003786,10**	**642,77**
EDFPLTF	9658,90	2,68	330879072,56	4063,39	**334870912,89**	**3987776,94**	**334,87**

100%							
Algoritmos	**Tempo (seg)**	**Tempo (horas)**	**Consumo (watts)**	**Cons. Ocioso (watts)**	**CONSUMO TOTAL**	**Rede (watts)**	**C. T. Mega Watts**
WQ	**98194,52**	**27,28**	**520465104,14**	**250,89**	**523522267,36**	**3056912,34**	**523,52**
WQR	**98287,42**	**27,30**	**612684569,75**	**112656,56**	**615854138,94**	**3056912,63**	**615,85**
Suff	**11049,69**	**3,07**	**372082951,33**	**5527,31**	**374273165,32**	**2184686,68**	**374,27**
XSuff	**189487,00**	**52,64**	**654411007,72**	**193962,87**	**657663313,88**	**3058343,28**	**657,66**
DFPLTF	**9959,11**	**2,77**	**382821488,39**	**4327,22**	**387412754,17**	**4586938,55**	**387,41**
LECSA	**39278,84**	**10,91**	**414810312,30**	**7373,79**	**418751001,05**	**3933314,96**	**418,75**
LECSA2	**39278,84**	**10,91**	**417883214,92**	**4886,14**	**421820968,05**	**3932866,99**	**421,82**
LECSA3	**39279,29**	**10,91**	**414810312,30**	**6690,27**	**418749537,13**	**3932534,55**	**418,75**
DLECSA	**91525,45**	**25,42**	**471289718,54**	**10669,54**	**476542991,31**	**5242603,23**	**476,54**
DLECSA2	**91527,89**	**25,42**	**484141580,84**	**10099,71**	**488083949,57**	**3932269,02**	**488,08**
DLECSA3	**91525,90**	**25,42**	**471769724,88**	**7538,05**	**473962682,97**	**2185420,04**	**473,96**
ECI	**15382,34**	**4,27**	**358410468,65**	**11158,44**	**363227505,21**	**4805878,11**	**363,23**
EWQ	**91708,36**	**25,47**	**472230611,83**	**231,12**	**476597783,11**	**4366940,15**	**476,60**
ESUFF	**11918,18**	**3,31**	**351095407,95**	**5979,67**	**356562283,73**	**5460896,11**	**356,56**
EXSUFF	**189487,00**	**52,64**	**654411007,72**	**186908,45**	**657656266,96**	**3058350,79**	**657,66**
EDFPLTF	**9988,30**	**2,77**	**330677899,21**	**4422,59**	**335270804,13**	**4588482,32**	**335,27**

Plataforma 2 – 29 tarefas

0%							
Algoritmos	**Tempo (seg)**	**Tempo (horas)**	**Consumo (watts)**	**Cons. Ocioso (watts)**	**CONSUMO TOTAL**	**Rede (watts)**	**C. T. Mega Watts**
WQ	326705,65	90,75	695933895,52	246,80	**695934589,42**	**447,09**	**695,93**
WQR	130833,88	36,34	827005469,69	156363,27	**827162292,01**	**459,04**	**827,16**
Suff	18775,19	5,22	337656395,76	15158,65	**337676457,73**	**4903,32**	**337,68**
XSuff	326635,61	90,73	695933895,52	335732,51	**696275975,56**	**6347,52**	**696,28**
DFPLTF	18775,19	5,22	337656395,76	15158,65	**337676457,73**	**4903,32**	**337,68**
LECSA	20336,88	5,65	79437131,18	56,22	**79437199,43**	**12,02**	**79,44**
LECSA2	20336,88	5,65	79437131,18	56,22	**79437199,43**	**12,02**	**79,44**
LECSA3	17564,69	4,88	81737505,84	51,79	**81737569,04**	**11,41**	**81,74**
DLECSA	101309,93	28,14	329096686,34	1874,56	**329098573,01**	**12,12**	**329,10**
DLECSA2	101309,93	28,14	329096686,34	1874,56	**329098573,01**	**12,12**	**329,10**
DLECSA3	87453,36	24,29	338626809,90	1699,48	**338628520,90**	**11,52**	**338,63**
ECI	20326,26	5,65	329471693,42	16079,31	**329497798,34**	**10025,61**	**329,50**
EWQ	20380,82	5,66	329148222,04	227,37	**329148883,09**	**433,68**	**329,15**
ESUFF	20326,26	5,65	329148222,04	16084,65	**329174320,75**	**10014,06**	**329,17**
EXSUFF	326635,61	90,73	695933895,52	335732,51	**696275975,56**	**6347,52**	**696,28**
EDFPLTF	20326,26	5,65	329148222,04	16084,65	**329174320,75**	**10014,06**	**329,17**

25%							
Algoritmos	**Tempo (seg)**	**Tempo (horas)**	**Consumo (watts)**	**Cons. Ocioso (watts)**	**CONSUMO TOTAL**	**Rede (watts)**	**C. T. Mega Watts**
WQ	367523,65	102,09	728505000,14	246,81	**728505692,84**	**445,89**	**728,51**
WQR	146341,88	40,65	873989944,44	175857,69	**874166261,44**	**459,30**	**874,17**
Suff	21093,29	5,86	337753760,65	17762,87	**337776527,69**	**5004,17**	**337,78**
XSuff	367452,82	102,07	728505000,14	378770,14	**728890305,27**	**6535,00**	**728,89**
DFPLTF	19746,64	5,49	337767669,92	16387,35	**337789280,53**	**5223,27**	**337,79**
LECSA	22867,24	6,35	89366772,58	56,22	**89366842,33**	**13,52**	**89,37**
LECSA2	22867,24	6,35	89366772,58	56,22	**89366842,33**	**13,52**	**89,37**
LECSA3	19748,48	5,49	91954694,07	51,79	**91954758,69**	**12,84**	**91,95**
DLECSA	108990,70	30,28	329096686,34	2603,64	**329099301,82**	**11,85**	**329,10**
DLECSA2	108990,70	30,28	329096686,34	2603,64	**329099301,82**	**11,85**	**329,10**
DLECSA3	94082,92	26,13	338626809,90	2359,97	**338629181,05**	**11,18**	**338,63**
ECI	22856,62	6,35	329498195,05	19159,85	**329527792,03**	**10437,13**	**329,53**
EWQ	22901,82	6,36	329147552,07	226,93	**329148214,66**	**435,65**	**329,15**
ESUFF	22856,62	6,35	329147552,07	19165,67	**329177151,95**	**10434,20**	**329,18**
EXSUFF	367452,82	102,07	728505000,14	378770,14	**728890305,27**	**6535,00**	**728,89**
EDFPLTF	22856,62	6,35	329142960,24	18958,12	**329172887,84**	**10969,48**	**329,17**

50%							
Algoritmos	**Tempo (seg)**	**Tempo (horas)**	**Consumo (watts)**	**Cons. Ocioso (watts)**	**CONSUMO TOTAL**	**Rede (watts)**	**C. T. Mega Watts**
WQ	392014,65	108,89	720729564,22	246,80	**720730256,80**	**445,78**	**720,73**
WQR	144214,25	40,06	812132295,25	173191,47	**812305947,28**	**460,56**	**812,31**
Suff	23439,36	6,51	337633752,76	20924,00	**337659592,52**	**4915,76**	**337,66**
XSuff	391943,42	108,87	720729564,22	406002,11	**721141928,12**	**6361,78**	**721,14**
DFPLTF	21930,43	6,09	337898999,30	19098,80	**337923622,41**	**5524,31**	**337,92**
LECSA	25397,61	7,05	187925556,06	340,69	**187925953,26**	**56,51**	**187,93**
LECSA2	25397,61	7,05	187925556,06	340,69	**187925953,26**	**56,51**	**187,93**
LECSA3	21932,27	6,09	193367585,24	310,76	**193367953,47**	**57,47**	**193,37**
DLECSA	66296,46	18,42	329096686,34	3614,36	**329100347,22**	**46,52**	**329,10**
DLECSA2	66296,46	18,42	329096686,34	3614,36	**329100347,22**	**46,52**	**329,10**
DLECSA3	60370,66	16,77	338626809,90	5351,00	**338632208,79**	**47,90**	**338,63**
ECI	25386,99	7,05	329536130,02	22177,20	**329568612,23**	**10305,02**	**329,57**
EWQ	25427,82	7,06	329147964,36	226,90	**329148623,91**	**432,64**	**329,15**
ESUFF	25386,99	7,05	329147964,36	22183,66	**329180430,90**	**10282,87**	**329,18**
EXSUFF	391943,42	108,87	720729564,22	406002,11	**721141928,12**	**6361,78**	**721,14**
EDFPLTF	25386,99	7,05	329136471,90	21809,03	**329170163,17**	**11882,24**	**329,17**

75%							
Algoritmos	**Tempo (seg)**	**Tempo (horas)**	**Consumo (watts)**	**Cons. Ocioso (watts)**	**CONSUMO TOTAL**	**Rede (watts)**	**C. T. Mega Watts**
WQ	449126,65	124,76	705612123,62	246,84	**705612818,10**	**447,64**	**705,61**
WQR	102626,11	28,51	598995824,20	121228,03	**599117516,17**	**463,94**	**599,12**
Suff	24731,35	6,87	337687125,54	22415,95	**337714457,80**	**4916,31**	**337,71**
XSuff	449087,53	124,75	705612123,62	466980,33	**706085443,92**	**6339,97**	**706,09**
DFPLTF	24114,22	6,70	338020301,06	21809,84	**338047960,83**	**5849,92**	**338,05**
LECSA	27927,24	7,76	272363026,35	1579,48	**272364734,33**	**128,50**	**272,36**
LECSA2	27927,20	7,76	272370050,67	1533,19	**272371709,31**	**125,45**	**272,37**
LECSA3	24116,05	6,70	280242877,16	1392,32	**280244400,89**	**131,42**	**280,24**
DLECSA	55761,81	15,49	329108663,23	8626,76	**329117398,64**	**108,65**	**329,12**
DLECSA2	47465,68	13,18	329114079,64	4457,83	**329118647,58**	**110,11**	**329,12**
DLECSA3	48138,86	13,37	338626809,90	7740,00	**338634666,39**	**116,49**	**338,63**
ECI	27917,35	7,75	329592572,59	25254,10	**329628169,37**	**10342,67**	**329,63**
EWQ	27962,82	7,77	329147799,45	226,93	**329148453,04**	**426,67**	**329,15**
ESUFF	27917,35	7,75	329147799,45	25261,45	**329183357,60**	**10296,71**	**329,18**
EXSUFF	449087,53	124,75	705612123,62	466980,33	**706085443,92**	**6339,97**	**706,09**
EDFPLTF	27917,35	7,75	329129602,19	24681,53	**329167120,85**	**12837,13**	**329,17**

100%							
Algoritmos	**Tempo (seg)**	**Tempo (horas)**	**Consumo (watts)**	**Cons. Ocioso (watts)**	**CONSUMO TOTAL**	**Rede (watts)**	**C. T. Mega Watts**
WQ	489976,65	136,10	787846689,98	246,82	**787847393,47**	**456,66**	**787,85**
WQR	153686,25	42,69	878190792,80	185080,72	**878376347,85**	**474,33**	**878,38**
Suff	28089,56	7,80	337617579,20	26317,61	**337648904,43**	**5007,62**	**337,65**
XSuff	489905,80	136,08	787846689,98	475138,61	**788328356,60**	**6528,01**	**788,33**
DFPLTF	26298,00	7,31	338109255,69	24519,55	**338139848,01**	**6072,76**	**338,14**
LECSA	30457,60	8,46	329128123,12	4683,54	**329133035,62**	**228,96**	**329,13**
LECSA2	28078,90	7,80	329167702,53	3756,52	**329171681,42**	**222,36**	**329,17**
LECSA3	28078,47	7,80	336718328,76	4736,89	**336723303,00**	**237,35**	**336,72**
DLECSA	30457,46	8,46	329146882,11	4738,98	**329151838,65**	**217,56**	**329,15**
DLECSA2	30467,46	8,46	329145645,25	4576,03	**329150445,02**	**223,74**	**329,15**
DLECSA3	28050,01	7,79	337514068,35	3714,71	**337518000,93**	**217,87**	**337,52**
ECI	30447,72	8,46	329146882,11	28053,10	**329185479,15**	**10543,94**	**329,19**
EWQ	30486,82	8,47	329146882,11	226,93	**329147541,18**	**432,14**	**329,15**
ESUFF	30447,72	8,46	329146882,11	28053,15	**329185466,73**	**10531,47**	**329,19**
EXSUFF	489905,80	136,08	787846689,98	475138,61	**788328356,60**	**6528,01**	**788,33**
EDFPLTF	30447,72	8,46	329128123,12	27694,87	**329169411,82**	**13593,83**	**329,17**

Plataforma 3 – 720 tarefas

0%

Algoritmos	Tempo (seg)	Tempo (horas)	Consumo (watts)	Cons. Ocioso (watts)	CONSUMO TOTAL	Rede (watts)	C. T. Mega Watts
WQ	7306,69	2,03	892401846,64	274,56	**894585550,27**	**2183429,07**	**894,59**
WQR	7396,82	2,05	896487148,75	1177,31	**898671755,19**	**2183429,14**	**898,67**
Suff	8039,09	2,23	918393875,69	2605,35	**920580263,12**	**2183782,08**	**920,58**
XSuff	23638,03	6,57	1176656911,67	15145,87	**1180165756,38**	**3493698,84**	**1180,17**
DFPLTF	14990,34	4,16	1089062864,35	7120,47	**1092563785,30**	**3493800,48**	**1092,56**
LECSA	29539,68	8,21	303258684,45	855,87	**303259547,17**	**6,85**	**303,26**
LECSA2	29539,68	8,21	303258684,45	855,89	**303259547,24**	**6,90**	**303,26**
LECSA3	16681,92	4,63	886510967,64	106,07	**886511082,02**	**8,31**	**886,51**
DLECSA	29684,34	8,25	302793206,90	977,13	**302794190,82**	**6,79**	**302,79**
DLECSA2	32531,95	9,04	295873488,49	1975,95	**295875471,16**	**6,72**	**295,88**
DLECSA3	16681,92	4,63	886510967,64	106,07	**886511082,02**	**8,31**	**886,51**
ECI	23518,53	6,53	892655247,34	23645,34	**897045917,72**	**4367025,04**	**897,05**
EWQ	7284,57	2,02	892104872,02	314,38	**894288615,84**	**2183429,44**	**894,29**
ESUFF	14745,76	4,10	602977669,90	11824,79	**603863079,23**	**873584,55**	**603,86**
EXSUFF	30554,94	8,49	1176656911,67	23675,17	**1179737575,85**	**3056989,02**	**1179,74**
EDFPLTF	14866,20	4,13	873206840,79	9305,17	**875399754,98**	**2183609,02**	**875,40**

25%

Algoritmos	Tempo (seg)	Tempo (horas)	Consumo (watts)	Cons. Ocioso (watts)	CONSUMO TOTAL	Rede (watts)	C. T. Mega Watts
WQ	8694,51	2,42	907435475,51	273,83	**909534026,80**	**2098277,46**	**909,53**
WQR	8786,18	2,44	911432208,69	2628,68	**913533114,84**	**2098277,48**	**913,53**
Suff	9704,83	2,70	1038840233,81	3759,50	**1041719892,26**	**2875898,95**	**1041,72**
XSuff	30125,21	8,37	1174634567,71	22829,43	**1178229696,61**	**3572299,48**	**1178,23**
DFPLTF	15576,12	4,33	1067487389,51	7600,75	**1069940818,96**	**2445828,69**	**1069,94**
LECSA	29521,10	8,20	301034613,23	975,59	**301035595,67**	**6,85**	**301,04**
LECSA2	29521,10	8,20	301034661,67	979,89	**301035648,48**	**6,92**	**301,04**
LECSA3	18418,33	5,12	876139659,43	519,58	**876140187,05**	**8,04**	**876,14**
DLECSA	32778,32	9,11	297473421,52	1923,70	**297475351,88**	**6,66**	**297,48**
DLECSA2	32877,78	9,13	298202262,09	1918,09	**298204186,77**	**6,59**	**298,20**
DLECSA3	20668,41	5,74	884210643,70	1097,69	**884211749,15**	**7,75**	**884,21**
ECI	23263,89	6,46	891779297,86	23328,54	**896198039,37**	**4395412,97**	**896,20**
EWQ	8668,21	2,41	896718489,15	289,43	**898825790,09**	**2107011,51**	**898,83**
ESUFF	17918,19	4,98	669327411,45	15962,58	**670566302,21**	**1222928,19**	**670,57**
EXSUFF	38569,21	10,71	1174634567,71	34372,46	**1178241206,40**	**3572266,23**	**1178,24**
EDFPLTF	15678,52	4,36	870344515,02	10036,71	**872422468,71**	**2067916,98**	**872,42**

50%							
Algoritmos	**Tempo (seg)**	**Tempo (horas)**	**Consumo (watts)**	**Cons. Ocioso (watts)**	**CONSUMO TOTAL**	**Rede (watts)**	**C. T. Mega Watts**
WQ	8527,49	2,37	905962466,44	273,25	**908146168,65**	**2183428,96**	**908,15**
WQR	8619,81	2,39	914630017,28	2366,02	**916815812,30**	**2183429,00**	**916,82**
Suff	9313,93	2,59	1009054138,79	3600,62	**1011656331,82**	**2598592,40**	**1011,66**
XSuff	29899,24	8,31	1176508413,64	23175,36	**1180047107,34**	**3515518,34**	**1180,05**
DFPLTF	14932,11	4,15	1072966999,91	7130,37	**1075660186,92**	**2686056,64**	**1075,66**
LECSA	16307,73	4,53	421158556,18	2457,17	**421161043,67**	**30,32**	**421,16**
LECSA2	12937,38	3,59	486410117,20	852,18	**486411001,75**	**32,37**	**486,41**
LECSA3	16307,73	4,53	552034461,97	2964,81	**552037458,53**	**31,74**	**552,04**
DLECSA	14859,20	4,13	479886500,00	2062,47	**479888591,43**	**28,96**	**479,89**
DLECSA2	15421,82	4,28	481264740,01	2284,19	**481267053,27**	**29,06**	**481,27**
DLECSA3	14303,90	3,97	625217948,64	2136,53	**625220116,61**	**31,45**	**625,22**
ECI	23412,43	6,50	883732433,35	23531,03	**887860983,04**	**4105018,66**	**887,86**
EWQ	8757,71	2,43	901620397,16	290,54	**903956953,24**	**2336265,54**	**903,96**
ESUFF	17742,67	4,93	652718663,65	14829,56	**653781746,71**	**1048253,50**	**653,78**
EXSUFF	38246,71	10,62	1176508413,64	33849,55	**1180057753,35**	**3515490,15**	**1180,06**
EDFPLTF	14466,36	4,02	840232220,18	8762,67	**842140818,75**	**1899835,90**	**842,14**

75%							
Algoritmos	**Tempo (seg)**	**Tempo (horas)**	**Consumo (watts)**	**Cons. Ocioso (watts)**	**CONSUMO TOTAL**	**Rede (watts)**	**C. T. Mega Watts**
WQ	8743,51	2,43	916107190,10	273,89	**918356392,91**	**2248928,92**	**918,36**
WQR	8834,14	2,45	922661406,79	2530,99	**924912866,75**	**2248928,97**	**924,91**
Suff	11013,63	3,06	1007094353,08	5349,52	**1009567286,21**	**2467583,60**	**1009,57**
XSuff	31410,93	8,73	1182296388,35	25517,57	**1186219503,59**	**3897597,68**	**1186,22**
DFPLTF	15494,02	4,30	1079483576,81	7697,43	**1082286547,92**	**2795273,68**	**1082,29**
LECSA	12357,04	3,43	672442852,16	2927,13	**674477612,93**	**2031833,64**	**674,48**
LECSA2	9249,06	2,57	763336620,36	1052,40	**765369514,42**	**2031841,67**	**765,37**
LECSA3	12356,39	3,43	672442852,16	2927,67	**674477613,14**	**2031833,31**	**674,48**
DLECSA	14052,04	3,90	763604857,99	3923,25	**765486193,71**	**1877412,48**	**765,49**
DLECSA2	11759,12	3,27	769464744,00	2552,39	**771669790,07**	**2202493,67**	**771,67**
DLECSA3	13095,58	3,64	762180555,70	3522,02	**764394696,36**	**2210618,65**	**764,39**
ECI	22881,92	6,36	875397354,30	22848,97	**879841807,55**	**4421604,29**	**879,84**
EWQ	8415,14	2,34	904493789,09	289,73	**906743008,22**	**2248929,39**	**906,74**
ESUFF	16978,51	4,72	670785348,71	14282,60	**671771462,34**	**971831,03**	**671,77**
EXSUFF	37447,96	10,40	1182296388,35	32880,46	**1186226845,84**	**3897577,03**	**1186,23**
EDFPLTF	15229,68	4,23	864391461,57	9355,07	**866136928,63**	**1736112,00**	**866,14**

100%							
Algoritmos	**Tempo (seg)**	**Tempo (horas)**	**Consumo (watts)**	**Cons. Ocioso (watts)**	**CONSUMO TOTAL**	**Rede (watts)**	**C. T. Mega Watts**
WQ	9148,20	2,54	911694922,39	274,01	**913747622,77**	**2052426,36**	**913,75**
WQR	9239,22	2,57	926075157,65	3093,12	**928130677,24**	**2052426,48**	**928,13**
Suff	10538,79	2,93	969495617,94	4687,85	**972208054,36**	**2707748,57**	**972,21**
XSuff	34762,32	9,66	1165679274,77	29344,70	**1169464281,78**	**3755662,31**	**1169,46**
DFPLTF	15105,74	4,20	1074889464,81	7241,20	**1077823030,99**	**2926324,99**	**1077,82**
LECSA	10924,20	3,03	891694865,01	3732,60	**894144374,68**	**2445777,07**	**894,14**
LECSA2	10924,20	3,03	965132762,27	3344,31	**967582131,13**	**2446024,56**	**967,58**
LECSA3	10924,72	3,03	891694865,01	3732,64	**894144556,95**	**2445959,31**	**894,14**
DLECSA	18795,97	5,22	1025868300,70	11506,09	**1028500157,13**	**2620350,35**	**1028,50**
DLECSA2	13042,53	3,62	985562106,97	4927,47	**988318581,05**	**2751546,61**	**988,32**
DLECSA3	13841,32	3,84	980430216,63	5903,53	**983318640,78**	**2882520,62**	**983,32**
ECI	25216,14	7,00	845537536,58	25915,99	**849930470,03**	**4367017,46**	**849,93**
EWQ	8731,39	2,43	907824564,43	289,18	**910051951,01**	**2227097,40**	**910,05**
ESUFF	17022,33	4,73	631087793,82	14447,00	**631975822,09**	**873581,27**	**631,98**
EXSUFF	42421,68	11,78	1165679274,77	39240,75	**1169474167,27**	**3755651,76**	**1169,47**
EDFPLTF	14884,86	4,13	839354790,04	8838,94	**841329034,96**	**1965405,98**	**841,33**

Plataforma 3 – 144 tarefas

0%							
Algoritmos	**Tempo (seg)**	**Tempo (horas)**	**Consumo (watts)**	**Cons. Ocioso (watts)**	**CONSUMO TOTAL**	**Rede (watts)**	**C. T. Mega Watts**
WQ	10885,93	3,02	945292481,05	253,90	**947476299,30**	**2183564,35**	**947,48**
WQR	10977,61	3,05	1079540376,01	3588,35	**1081727529,43**	**2183565,07**	**1081,73**
Suff	9698,86	2,69	763519196,83	7290,26	**765711798,29**	**2185311,20**	**765,71**
XSuff	21402,57	5,95	1128374120,58	13894,68	**1130572870,07**	**2184854,81**	**1130,57**
DFPLTF	10393,68	2,89	754476280,55	7874,23	**756669465,00**	**2185310,22**	**756,67**
LECSA	28441,74	7,90	301446944,33	679,06	**301447649,21**	**25,82**	**301,45**
LECSA2	28439,11	7,90	301447975,04	586,88	**301448587,78**	**25,85**	**301,45**
LECSA3	17633,35	4,90	879037834,05	308,18	**879038169,89**	**27,66**	**879,04**
DLECSA	27507,57	7,64	300109471,97	659,06	**300110156,79**	**25,76**	**300,11**
DLECSA2	32916,05	9,14	297090248,67	1914,74	**297092189,16**	**25,75**	**297,09**
DLECSA3	17633,35	4,90	879037834,05	308,18	**879038169,89**	**27,66**	**879,04**
ECI	31997,45	8,89	903414522,12	34248,33	**907816882,62**	**4368112,17**	**907,82**
EWQ	10891,93	3,03	945292481,05	272,45	**947476319,14**	**2183565,64**	**947,48**
ESUFF	18532,46	5,15	410848783,66	18784,16	**410868687,46**	**1119,63**	**410,87**
EXSUFF	42555,48	11,82	1128374120,58	36521,75	**1130595327,23**	**2184684,90**	**1130,60**
EDFPLTF	17921,48	4,98	423580627,21	17560,98	**423599307,73**	**1119,54**	**423,60**

25%							
Algoritmos	**Tempo (seg)**	**Tempo (horas)**	**Consumo (watts)**	**Cons. Ocioso (watts)**	**CONSUMO TOTAL**	**Rede (watts)**	**C. T. Mega Watts**
WQ	12195,26	3,39	949353883,78	253,57	**951712373,63**	**2358236,27**	**951,71**
WQR	12286,97	3,41	993125984,41	5539,06	**995489760,10**	**2358236,63**	**995,49**
Suff	11998,32	3,33	822636917,06	10177,30	**824559433,50**	**1912339,14**	**824,56**
XSuff	35004,70	9,72	1120979010,29	31123,39	**1123369684,57**	**2359550,89**	**1123,37**
DFPLTF	9573,79	2,66	706419558,65	7002,80	**708339176,44**	**1912614,99**	**708,34**
LECSA	28603,31	7,95	298664389,14	898,15	**298665313,09**	**25,81**	**298,67**
LECSA2	27872,31	7,74	298665852,75	664,18	**298666542,77**	**25,84**	**298,67**
LECSA3	19484,69	5,41	866485304,98	740,47	**866486073,15**	**27,71**	**866,49**
DLECSA	29521,55	8,20	297233402,83	1200,52	**297234629,87**	**26,52**	**297,23**
DLECSA2	31785,76	8,83	298914762,71	1595,35	**298916384,40**	**26,34**	**298,92**
DLECSA3	21545,12	5,98	878652500,60	1285,57	**878653813,85**	**27,68**	**878,65**
ECI	32156,60	8,93	900683215,37	34444,12	**904725553,54**	**4007894,04**	**904,73**
EWQ	12214,39	3,39	963568668,03	269,84	**965839841,28**	**2270903,41**	**965,84**
ESUFF	23565,95	6,55	455358889,43	24157,68	**455384209,57**	**1162,46**	**455,38**
EXSUFF	43280,47	12,02	1120979010,29	38908,19	**1123377305,68**	**2359387,20**	**1123,38**
EDFPLTF	17854,67	4,96	423073571,47	17227,40	**423092052,54**	**1253,67**	**423,09**

50%							
Algoritmos	**Tempo (seg)**	**Tempo (horas)**	**Consumo (watts)**	**Cons. Ocioso (watts)**	**CONSUMO TOTAL**	**Rede (watts)**	**C. T. Mega Watts**
WQ	13515,19	3,75	948676701,02	253,59	**950860517,57**	**2183562,95**	**950,86**
WQR	13607,05	3,78	992586637,98	7266,55	**994777467,93**	**2183563,40**	**994,78**
Suff	17506,08	4,86	855324390,14	16158,58	**856979888,82**	**1639340,10**	**856,98**
XSuff	38979,74	10,83	1110115612,10	31506,51	**1112331936,56**	**2184817,95**	**1112,33**
DFPLTF	10717,19	2,98	719122321,87	8369,17	**720770670,25**	**1639979,21**	**720,77**
LECSA	17564,92	4,88	408456482,95	3253,74	**408459862,00**	**125,31**	**408,46**
LECSA2	12406,89	3,45	501508538,18	858,80	**501509524,34**	**127,36**	**501,51**
LECSA3	16866,10	4,69	527841582,10	3293,04	**527845005,96**	**130,82**	**527,85**
DLECSA	15923,30	4,42	461182690,14	2219,12	**461185028,97**	**119,72**	**461,19**
DLECSA2	16652,36	4,63	486279002,09	2699,74	**486281823,75**	**121,92**	**486,28**
DLECSA3	14574,77	4,05	638692883,08	2284,13	**638695292,43**	**125,21**	**638,70**
ECI	30936,44	8,59	901690903,78	32953,07	**906091979,54**	**4368122,69**	**906,09**
EWQ	13536,84	3,76	961119605,26	269,88	**962757595,95**	**1637720,82**	**962,76**
ESUFF	22636,21	6,29	435832300,40	24386,35	**435857807,60**	**1120,85**	**435,86**
EXSUFF	52198,58	14,50	1110115612,10	47487,57	**1112347786,36**	**2184686,68**	**1112,35**
EDFPLTF	17686,18	4,91	420130288,88	16898,12	**420148637,46**	**1450,46**	**420,15**

75%							
Algoritmos	**Tempo (seg)**	**Tempo (horas)**	**Consumo (watts)**	**Cons. Ocioso (watts)**	**CONSUMO TOTAL**	**Rede (watts)**	**C. T. Mega Watts**
WQ	14841,03	4,12	1027827343,59	253,47	**1030829933,05**	**3002335,99**	**1030,83**
WQR	14933,19	4,15	1069459624,06	8467,48	**1072470427,88**	**3002336,35**	**1072,47**
Suff	18814,90	5,23	867543304,68	18170,21	**868927953,78**	**1366478,89**	**868,93**
XSuff	43862,95	12,18	1127951383,04	34250,26	**1130989234,74**	**3003601,44**	**1130,99**
DFPLTF	11589,10	3,22	740838341,12	8502,63	**742214100,47**	**1367256,72**	**742,21**
LECSA	15398,96	4,28	649503078,09	5230,39	**651540426,00**	**2032117,51**	**651,54**
LECSA2	10251,44	2,85	760365908,11	1799,63	**762399846,36**	**2032138,62**	**762,40**
LECSA3	15400,08	4,28	649503078,09	5119,27	**649711408,90**	**203211,55**	**649,71**
DLECSA	15057,61	4,18	721975983,70	3262,65	**723605003,48**	**1625757,13**	**723,61**
DLECSA2	12387,40	3,44	718566151,48	3339,59	**720195260,08**	**1625769,01**	**720,20**
DLECSA3	16049,78	4,46	672005537,95	3749,48	**673635040,10**	**1625752,67**	**673,64**
ECI	28018,67	7,78	800415328,54	29621,44	**802848113,39**	**2403163,41**	**802,85**
EWQ	14849,77	4,12	999237149,59	269,71	**1001420991,84**	**2183572,53**	**1001,42**
ESUFF	22564,00	6,27	430794380,87	23048,87	**430818611,16**	**1181,42**	**430,82**
EXSUFF	45879,85	12,74	1127951383,04	37660,21	**1130992554,98**	**3003511,74**	**1130,99**
EDFPLTF	17716,48	4,92	434850133,44	16162,53	**434867878,31**	**1582,35**	**434,87**

100%							
Algoritmos	**Tempo (seg)**	**Tempo (horas)**	**Consumo (watts)**	**Cons. Ocioso (watts)**	**CONSUMO TOTAL**	**Rede (watts)**	**C. T. Mega Watts**
WQ	16174,05	4,49	985344537,19	253,48	**988401704,12**	**3056913,44**	**988,40**
WQR	16264,99	4,52	1001234412,13	10510,99	**1004301836,77**	**3056913,65**	**1004,30**
Suff	16970,47	4,71	930039447,69	13246,42	**931364540,38**	**1311846,28**	**931,36**
XSuff	42625,85	11,84	1110559541,04	40692,28	**1113658374,98**	**3058141,65**	**1113,66**
DFPLTF	11354,08	3,15	806253659,54	7891,68	**809321190,61**	**3059639,39**	**809,32**
LECSA	13452,21	3,74	730467279,01	8586,97	**732006093,03**	**1530227,05**	**732,01**
LECSA2	8106,94	2,25	856849673,21	2149,25	**858383268,22**	**1531445,76**	**858,38**
LECSA3	13450,74	3,74	730467279,01	8505,75	**732006920,01**	**1531135,26**	**732,01**
DLECSA	16008,95	4,45	935394505,13	8366,42	**937587751,93**	**2184880,38**	**937,59**
DLECSA2	16008,95	4,45	980622747,02	5769,79	**982814357,53**	**2185840,71**	**982,81**
DLECSA3	16008,95	4,45	935526096,87	6103,13	**937717910,96**	**2185710,96**	**937,72**
ECI	21387,37	5,94	819348663,05	21148,70	**822427933,15**	**3058121,40**	**822,43**
EWQ	16165,14	4,49	982833758,29	269,82	**985017593,33**	**2183565,22**	**985,02**
ESUFF	21548,93	5,99	418721201,73	22382,68	**420927149,63**	**2183565,22**	**420,93**
EXSUFF	52249,81	14,51	1110559541,04	49564,15	**1113667200,91**	**3058095,72**	**1113,67**
EDFPLTF	18737,95	5,20	425193035,20	17623,25	**425212428,48**	**1770,04**	**425,21**

Plataforma 3 – 29 tarefas

0%

Algoritmos	Tempo (seg)	Tempo (horas)	Consumo (watts)	Cons. Ocioso (watts)	CONSUMO TOTAL	Rede (watts)	C. T. Mega Watts
WQ	53207,65	14,78	983513834,94	249,86	**983514531,89**	**447,09**	**983,51**
WQR	53299,04	14,81	1074917156,03	58119,82	**1074975734,77**	**458,91**	**1074,98**
Suff	12898,43	3,58	767761996,16	11631,41	**767782182,99**	**8555,42**	**767,78**
XSuff	53144,22	14,76	983513834,94	56717,77	**983576900,24**	**6347,52**	**983,58**
DFPLTF	12898,43	3,58	767761996,16	11631,41	**767782182,99**	**8555,42**	**767,78**
LECSA	17563,35	4,88	63845311,93	147,77	**63845469,70**	**10,00**	**63,85**
LECSA2	17563,35	4,88	63845311,93	68,27	**63845390,19**	**9,99**	**63,85**
LECSA3	12402,57	3,45	163704396,63	73,45	**163704481,34**	**11,26**	**163,70**
DLECSA	64053,17	17,79	264502006,57	423,94	**264502440,63**	**10,12**	**264,50**
DLECSA2	64052,81	17,79	264502006,57	450,44	**264502467,12**	**10,12**	**264,50**
DLECSA3	61639,90	17,12	678203928,91	1078,95	**678205019,27**	**11,41**	**678,21**
ECI	22367,72	6,21	315493328,58	22212,07	**315520829,68**	**5289,04**	**315,52**
EWQ	22418,92	6,23	313018540,11	266,03	**313019219,39**	**413,25**	**313,02**
ESUFF	22368,24	6,21	313018540,11	22329,95	**313046222,08**	**5352,02**	**313,05**
EXSUFF	53144,22	14,76	983513834,94	56717,77	**983576900,24**	**6347,52**	**983,58**
EDFPLTF	22368,24	6,21	313018540,11	22329,95	**313046222,08**	**5352,02**	**313,05**

25%

Algoritmos	Tempo (seg)	Tempo (horas)	Consumo (watts)	Cons. Ocioso (watts)	CONSUMO TOTAL	Rede (watts)	C. T. Mega Watts
WQ	59779,65	16,61	993894776,68	249,90	**993895472,48**	**445,89**	**993,90**
WQR	59627,99	16,56	1010005453,67	66263,92	**1010072176,70**	**459,12**	**1010,07**
Suff	14432,43	4,01	768138295,88	13224,47	**768160435,95**	**8915,59**	**768,16**
XSuff	59775,36	16,60	993894776,68	65097,59	**993966409,27**	**6535,00**	**993,97**
DFPLTF	14115,39	3,92	773544429,68	12976,73	**773566709,97**	**9303,56**	**773,57**
LECSA	19747,14	5,49	71825975,92	157,71	**71826144,89**	**11,25**	**71,83**
LECSA2	19747,14	5,49	71825975,92	68,27	**71826055,43**	**11,24**	**71,83**
LECSA3	13941,14	3,87	184167446,21	73,45	**184167532,33**	**12,66**	**184,17**
DLECSA	76781,36	21,33	264502006,57	790,37	**264502806,90**	**9,96**	**264,50**
DLECSA2	71804,57	19,95	264502006,57	526,08	**264502542,55**	**9,90**	**264,50**
DLECSA3	63978,08	17,77	678203928,91	1033,51	**678204973,63**	**11,21**	**678,20**
ECI	25150,45	6,99	315529086,27	25621,13	**315560101,64**	**5394,24**	**315,56**
EWQ	25197,92	7,00	312744949,25	265,72	**312745627,70**	**412,73**	**312,75**
ESUFF	25140,61	6,98	312744949,25	25402,19	**312775809,73**	**5458,30**	**312,78**
EXSUFF	59775,36	16,60	993894776,68	65097,59	**993966409,27**	**6535,00**	**993,97**
EDFPLTF	22366,55	6,21	310179859,18	22454,42	**310208039,05**	**5725,45**	**310,21**

50%							
Algoritmos	**Tempo (seg)**	**Tempo (horas)**	**Consumo (watts)**	**Cons. Ocioso (watts)**	**CONSUMO TOTAL**	**Rede (watts)**	**C. T. Mega Watts**
WQ	66457,65	18,46	979598235,05	249,84	**979598930,66**	**445,78**	**979,60**
WQR	66549,86	18,49	922432132,88	76197,69	**922508791,01**	**460,44**	**922,51**
Suff	16083,71	4,47	777244984,34	15272,88	**777268948,48**	**8691,26**	**777,27**
XSuff	66406,66	18,45	979598235,05	73161,58	**979677758,41**	**6361,78**	**979,68**
DFPLTF	15468,84	4,30	784930845,69	14477,06	**784955317,15**	**9994,40**	**784,96**
LECSA	23980,09	6,66	163598700,97	379,75	**163599133,78**	**53,07**	**163,60**
LECSA2	22800,86	6,33	163599216,33	106,47	**163599375,73**	**52,94**	**163,60**
LECSA3	15479,72	4,30	465745732,40	175,15	**465745966,73**	**59,17**	**465,75**
DLECSA	53048,08	14,74	279145183,57	2847,51	**279148075,95**	**44,87**	**279,15**
DLECSA2	49346,91	13,71	284540005,90	2098,34	**284542148,43**	**44,18**	**284,54**
DLECSA3	40527,36	11,26	813537707,44	1632,52	**813539389,05**	**49,08**	**813,54**
ECI	27886,33	7,75	316464732,33	28263,03	**316498339,45**	**5344,10**	**316,50**
EWQ	27978,92	7,77	313752482,00	265,72	**313753166,28**	**418,56**	**313,75**
ESUFF	27886,98	7,75	313752482,00	28059,46	**313786096,40**	**5554,94**	**313,79**
EXSUFF	66406,66	18,45	979598235,05	73161,58	**979677758,41**	**6361,78**	**979,68**
EDFPLTF	23985,98	6,66	307847309,51	24568,00	**307877968,81**	**6091,30**	**307,88**

75%							
Algoritmos	**Tempo (seg)**	**Tempo (horas)**	**Consumo (watts)**	**Cons. Ocioso (watts)**	**CONSUMO TOTAL**	**Rede (watts)**	**C. T. Mega Watts**
WQ	73041,65	20,29	972168415,36	249,85	**972169112,86**	**447,64**	**972,17**
WQR	48740,93	13,54	907231877,77	54668,70	**907287010,17**	**463,70**	**907,29**
Suff	17282,80	4,80	782171280,24	16657,64	**782196620,34**	**8682,47**	**782,20**
XSuff	73037,96	20,29	972168415,36	82799,28	**972257554,61**	**6339,97**	**972,26**
DFPLTF	17007,41	4,72	793332243,98	16192,86	**793359194,60**	**10757,75**	**793,36**
LECSA	26409,24	7,34	247129563,95	943,32	**247130634,00**	**126,73**	**247,13**
LECSA2	26004,39	7,22	247130986,33	656,68	**247131768,95**	**125,93**	**247,13**
LECSA3	17018,29	4,73	715300506,30	624,00	**715301268,36**	**138,06**	**715,30**
DLECSA	35261,80	9,79	294343426,17	2440,90	**294345979,52**	**112,45**	**294,35**
DLECSA2	35261,80	9,79	294342802,59	2441,36	**294345358,92**	**114,97**	**294,35**
DLECSA3	33942,72	9,43	838619412,04	3642,22	**838623174,23**	**119,96**	**838,62**
ECI	30706,14	8,53	316365389,11	32445,80	**316401477,14**	**3642,22**	**316,40**
EWQ	30758,92	8,54	313136969,75	265,69	**313137643,70**	**408,25**	**313,14**
ESUFF	30706,06	8,53	313136969,75	32653,07	**313175131,90**	**5509,07**	**313,18**
EXSUFF	73037,96	20,29	972168415,36	82799,28	**972257554,61**	**6339,97**	**972,26**
EDFPLTF	26415,13	7,34	305107300,50	27671,04	**305141449,01**	**6477,48**	**305,14**

100%							
Algoritmos	**Tempo (seg)**	**Tempo (horas)**	**Consumo (watts)**	**Cons. Ocioso (watts)**	**CONSUMO TOTAL**	**Rede (watts)**	**C. T. Mega Watts**
WQ	79673,65	22,13	971331425,68	249,91	**971332132,25**	**456,66**	**971,33**
WQR	59665,86	16,57	899057603,27	67838,63	**899125916,12**	**474,22**	**899,13**
Suff	19227,82	5,34	757282586,40	18643,99	**757310043,41**	**8813,02**	**757,31**
XSuff	79669,26	22,13	971331425,68	89881,79	**971427835,48**	**6528,01**	**971,43**
DFPLTF	18545,99	5,15	798909490,59	17925,78	**798938675,85**	**11259,48**	**798,94**
LECSA	24551,47	6,82	303872373,91	1671,09	**303874264,97**	**219,97**	**303,87**
LECSA2	22765,52	6,32	303872992,34	1094,81	**303874306,37**	**219,22**	**303,87**
LECSA3	19275,96	5,35	682304300,29	2053,32	**682306584,59**	**230,98**	**682,31**
DLECSA	33486,53	9,30	315130712,37	2461,96	**315133391,22**	**216,88**	**315,13**
DLECSA2	33486,53	9,30	315135762,87	2317,28	**315138291,21**	**211,06**	**315,14**
DLECSA3	19251,33	5,35	769343883,25	1531,56	**769345636,55**	**221,75**	**769,35**
ECI	33488,91	9,30	316550524,75	34474,13	**316590485,18**	**5486,30**	**316,59**
EWQ	33545,92	9,32	315130712,37	265,69	**315131404,40**	**426,34**	**315,13**
ESUFF	33499,14	9,31	315130712,37	35944,42	**315172285,43**	**5628,64**	**315,17**
EXSUFF	79669,26	22,13	971331425,68	89881,79	**971427835,48**	**6528,01**	**971,43**
EDFPLTF	24518,68	6,81	303872373,91	24826,71	**303903907,43**	**6706,81**	**303,90**