Cássio Henrique Volpatto Forte

Desenvolvimento e avaliação de um escalonador para grades colaborativas baseado em consumo de energia

São José do Rio Preto

2018

Cássio Henrique Volpatto Forte

Desenvolvimento e avaliação de um escalonador para grades colaborativas baseado em consumo de energia

Dissertação apresentada como parte dos requisitos para obtenção do título de Mestre em Ciência da Computação, junto ao Programa de Pós-Graduação em Ciência da Computação, do Instituto de Biociências, Letras e Ciências Exatas da Universidade Estadual Paulista "Júlio Mesquita Filho", Câmpus de São José do Rio Preto.

Financiadora: CAPES/DS

Universidade Estadual Paulista "Júlio de Mesquita Filho" (UNESP)

Instituto de Biociências, Letras e Ciências Exatas (IBILCE)

Programa de Pós-Graduação em Ciência da Computação (PPGCC)

Prof. Dr. Aleardo Manacero Jr.

São José do Rio Preto

2018

Forte, Cássio Henrique Volpatto.
Desenvolvimento e avaliação de um escalonador para grades colaborativas baseado em consumo de energia /Cássio Henrique Volpatto Forte. -- São José do Rio Preto, 2018
95 f. : il.

Orientador: Aleardo Manacero Jr.
Dissertação (mestrado) – Universidade Estadual Paulista "Júlio de Mesquita Filho", Instituto de Biociências, Letras e Ciências Exatas

1. Computação. 2. Computação em grade (Sistemas de computação) 3. Energia. I. Universidade Estadual Paulista "Júlio de Mesquita Filho". Instituto de Biociências, Letras e Ciências Exatas. II. Título.

CDU – 681.3.025

Ficha catalográfica elaborada pela Biblioteca do IBILCE
UNESP - Câmpus de São José do Rio Preto

Cássio Henrique Volpatto Forte

Desenvolvimento e avaliação de um escalonador para grades colaborativas baseado em consumo de energia

Dissertação apresentada como parte dos requisitos para obtenção do título de Mestre em Ciência da Computação, junto ao Programa de Pós-Graduação em Ciência da Computação, do Instituto de Biociências, Letras e Ciências Exatas da Universidade Estadual Paulista "Júlio Mesquita Filho", Câmpus de São José do Rio Preto.

Financiadora: CAPES/DS

Trabalho aprovado. São José do Rio Preto, 07 de Fevereiro de 2018:

Prof. Dr. Aleardo Manacero Jr.
UNESP - São José do Rio Preto
Orientador

Prof. Dr. Paulo Sérgio Lopes de Souza
USP - São Carlos

Prof. Dr. Edson Borin
UNICAMP - Campinas

São José do Rio Preto
2018

Aos meus pais Gislaine e Manuel, e à minha irmã Amanda.

Agradecimentos

Agradeço a todos que contribuíram com este projeto. Primeiramente agradeço a Deus pelas oportunidades da vida e pelos recursos necessários para poder aproveitá-las. Agradeço também à minha mãe, à minha irmã e a meu pai, que me dedicaram amor constante, e me auxiliaram em todos os aspectos da vida. Devo tudo a eles.

Quero agradecer também ao meu orientador, Professor Doutor Aleardo Manacero Junior, que mesmo em face de grandes dificuldades durante o projeto me deu total apoio e confiou em mim. Com a orientação do Professor Aleardo diversos obstáculos do projeto foram vencidos, permitindo que os trabalhos desenvolvidos gerassem frutos.

Quero agradecer também à Professora Doutora Renata Spolon Lobato, que também me deu apoio e orientação durante os trabalhos. Agradeço também à Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (Capes) pelo aporte financeiro, e ao Grupo de Sistemas Paralelos e Distribuídos (GSPD) da UNESP, campus de São José do Rio Preto, pelo acolhimento e infraestrutura. Agradeço também aos caros amigos do GSPD pela ajuda e pelas sugestões durante todo o projeto.

"Possuímos em nós mesmos, pelo pensamento e a vontade, um poder de ação que se estende muito além dos limites de nossa esfera corpórea"

-Allan Kardec

Resumo

A complexidade crescente das aplicações e o grande volume de dados utilizados por elas levam a um uso sempre crescente de sistemas distribuídos de alto desempenho. Nas últimas décadas o impacto do consumo de energia cresceu em relevância para o bom funcionamento desses sistemas, e seu tratamento é um grande desafio aos projetistas de *hardware*, desenvolvedores de aplicações e administradores. A dificuldade desse tratamento decorre do conflito entre consumo de energia e desempenho. Reduzir o consumo de energia das máquinas em um sistema distribuído causa prejuízos ao desempenho, enquanto fazer com que elas trabalhem mais rapidamente proporciona melhor desempenho mas causa aumento no consumo de energia. Nesse cenário, as políticas de escalonamento de tarefas podem levar em conta o consumo de energia, auxiliando no tratamento do problema. Este texto apresenta o desenvolvimento e avaliação de um novo algoritmo de escalonamento de tarefas independentes em grades computacionais federadas, o EHOSEP (*Energy-aware Heterogeneous Owner-Share Enforcement Policy*). O objetivo do novo algoritmo é tratar o consumo de energia, associando-o a um critério de justiça de propriedade. Esse critério de justiça decorre das chamadas grades federadas ou cooperativas, formadas por recursos computacionais de diferentes proprietários, procurando estimular seu compartilhamento pela garantia de uso justo. Os resultados obtidos com a simulação da aplicação do EHOSEP em diferentes modelos de grade mostram que é possível estimular o uso da grade atendendo-se limites de potência.

Palavras-chaves: grades federadas. energia. escalonamento.

Abstract

The increasing complexity of applications and the large volume of data used by them lead to an ever-increasing use of high-performance distributed systems. In recent decades the energy consumption is becoming more relevant to the proper functioning of these systems, and its management is a major challenge to hardware designers, application developers and administrators. The difficulty of this management arises from the conflict between power consumption and performance. Reducing energy consumption of machines in a distributed system reduces performance as well, while making machines work faster provides better performance at a cost of an increase in energy consumption. In this scenario, task scheduling policies may also consider energy consumption, helping to solve this problem. This document presents the development and evaluation of a new scheduling algorithm for independent tasks in federated computing grids, the EHOSEP (Energy-aware Heterogeneous Owner-Share Enforcement Policy). The goal of the new algorithm is to address energy consumption by associating it with a ownership fairness criterion. This fairness criterion stems from the so-called federated or cooperative grids, formed by computational resources of different owners, aiming the resource sharing by the guarantee of fair usage. Results achieved with the simulation of EHOSEP applied to different grid models show that it is possible to stimulate the use of the grid even limiting energy consumption.

Key-words: federated grids. energy. scheduling.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

API	*Application Programming Interface*
BoT	*Bag of Tasks*
CPU	*Central Processing Unit*
CRAC	*Computer Room Air Condicioning*
DVFS	*Dynamic Voltage and Frequency Scaling*
FLOPS	*Floating-point Operations Per Second*
GFLOPs	*Giga Floating-point Operations Per second*
GPU	*Graphics Processing Unit*
MFLOPS	*Mega Floating-point Operations Per Second*
MFLOP	*Mega Floating-point Operations*
NoC	*Network-on-Chip*
SLA	*Service Level Agreement*
SO	Sistema Operacional
VM	*Virtual Machine*
VPN	*Virtual Private Network*

Sumário

1 Introdução

Este trabalho propõe um algoritmo de escalonamento de tarefas em grades computacionais federadas que seja dirigido pela otimização energética do sistema, além da justiça em seu uso. Nas seções seguintes se descreve a relevância e motivação do problema, bem como o detalhamento do objetivo específico do trabalho.

1.1 Relevância

A necessidade por sistemas computacionais distribuídos de alto desempenho é sempre crescente. A razão para isso é o aumento na complexidade dos problemas resolvidos computacionalmente, tanto para maior precisão de resultados como para tratar um maior volume de dados para resolver esses problemas. Nesse cenário os projetistas desses sistemas preocupavam-se com aspectos relacionados a desempenho, como reduzir tempo de resposta, reduzir perdas de desempenho causadas por comunicação, otimizar o particionamento de problemas, etc. Entretanto, durante a última década a evolução dos sistemas utilizados gerou uma outra preocupação, que é a do consumo de energia.

O consumo de energia em sistemas de grande porte passou a ser um componente forte na parcela de custos operacionais. Apesar de o *hardware* dos sistemas computacionais estar evoluindo constantemente em relação à eficiência energética, o consumo final em sistemas de grande porte tem crescido em função do aumento no tamanho dos sistemas (1). Em particular, as grades computacionais são sistemas distribuídos fortemente relacionados à capacidade de expansão pela adição de novos recursos computacionais, e portanto apresentam potencial para conter grande número de elementos de processamento, levando a um grande consumo de energia.

Grades federadas ou cooperativas são grades em que os recursos computacionais pertencem a diferentes entidades. Nesses casos o gerenciamento é compartilhado, sendo necessário ter mecanismos que incentivem tanto o compartilhamento como o uso eficiente do sistema do ponto de vista energético. Como nessas redes os usuários também são os provedores de recursos, uma estratégia possível é criar regras de alocação que favoreçam os que compartilhem mais recursos energeticamente eficientes.

O compartilhamento de recursos computacionais em grades cooperativas requer o estabelecimento de regras e condições, que são definidas em SLA (*Service Level Agreement*), levando a critérios para alocação de tarefas. Nesse tipo de sistema o escalonamento de tarefas pode satisfazer critérios diversos, entre os quais aparecem com destaque aqueles que são baseados em justiça de acesso, vários deles baseados em histórico de uso.

Um outro critério de justiça, útil em grades federadas, consiste em considerar a propriedade de recursos, buscando garantir que usuários que ofereçam máquinas para compartilhamento consigam usar a sua parte sempre que quiserem. Uma proposta nesse sentido foi feita em trabalho prévio por Falavinha com o algoritmo OSEP (*Owner Share Enforcement Policy*) (2), com o objetivo de atrair provedores de recursos com a garantia de que as tarefas submetidas por eles serão atendidas por um número de máquinas maior ou igual ao oferecido pelo provedor à grade. Em ambientes de *hardware* homogêneo essa abordagem é suficiente para garantir o poder computacional da quota de cada participante da grade, uma vez que o poder computacional das máquinas é o mesmo.

Também em trabalho prévio Forte (3) estendeu o OSEP para grades heterogêneas, dando origem ao algoritmo HOSEP (*Heterogeneous Owner Share Enforcement Policy*). Em vez de considerar o número de máquinas que executa tarefas de um determinado usuário, o HOSEP leva em conta o poder de processamento dessas máquinas e, com isso, verifica se o poder computacional acumulado por meio delas é adequado, comparando-o com o poder total fornecido pelo usuário. Apesar de ter atingido os objetivos, o HOSEP não oferece nenhum tipo de tratamento para o consumo de energia, e a inclusão desse tratamento constitui a base para o trabalho apresentado nesta dissertação.

1.2 Motivação

O cenário descrito nos parágrafos anteriores indica que o uso de grades computacionais federadas, embora promova o compartilhamento de recursos entre diferentes grupos e entidades, também está sujeito ao elevado consumo de energia em sua operação. Esse compartilhamento leva a uma melhor ocupação dos nós do sistema, que são os recursos computacionais, e permite que os participantes obtenham mais desempenho sem comprar mais computadores. Neste cenário toda otimização na utilização dos nós é de interesse dos provedores de recursos.

Em grades federadas o compartilhamento é feito de forma que cada provedor pague pelo funcionamento dos recursos que fornece. Tipicamente não existe pagamento em dinheiro pelos recursos utilizados dos parceiros, o que significa que um usuário não pagará pela conta de energia elétrica dos recursos de outros provedores que ele venha a utilizar.

Com isso a identificação de uma política que permita fazer o uso eficiente da grade federada, tanto do ponto de vista energético quanto de justiça de desempenho é fundamental. Essa política conduziria a que potenciais usuários da grade provenham recursos e que esses recursos sejam energeticamente eficientes.

1.3 Objetivo

O objetivo deste trabalho é desenvolver um novo algoritmo de escalonamento de tarefas em grades computacionais federadas, isto é, grades em que os usuários também são proprietários dos nós compartilhados. Este novo escalonador é baseado em dois critérios, que são a redução de energia e a justiça de propriedade. Ele foi desenvolvido a partir do algoritmo HOSEP, recebendo a denominação de *Energy-aware Heterogeneous Owner-Share Enforcement Policy*, ou EHOSEP. A redução no consumo de energia é obtida estimulando os usuários a oferecerem máquinas energeticamente eficientes e com poder computacional relevante.

A justiça de propriedade segue a formulação do HOSEP, ou seja, estimula o compartilhamento pela garantia de usar o poder computacional proporcional ao oferecido. Dessa forma busca-se a formação de grades federadas eficientes e de elevado poder computacional. Considerando o ambiente típico de grades, as tarefas consideradas pelo algoritmo de escalonamento serão *Bag of Tasks* (4), ou seja, serão submetidos pelos usuários de forma independente, possivelmente em conjuntos de tarefas que ocuparão um ou mais nós do sistema durante sua execução. Dessa forma o EHOSEP atenderá a aplicações cuja execução está concentrada na CPU, executando por exemplo grande volume de cálculos matemáticos e operações lógicas.

1.4 Organização do texto

O texto está organizado em cinco capítulos contando esta Introdução. No Capítulo 2 é apresentada uma revisão bibliográfica abrangendo grades computacionais e seus algoritmos de escalonamento, concentrando-se naqueles que guardam alguma relação com o escalonador desenvolvido neste trabalho. No Capítulo 3 é apresentado em detalhes o algoritmo EHOSEP (*Energy-aware Heterogeneous Owner-Share Enforcement Policy*), que é o escalonador proposto. Os testes dele e seus respectivos resultados são mostrados no Capítulo 4, enquanto as conclusões do trabalho são apresentadas no Capítulo 5. Deve ser observado ainda que parte dos resultados é apresentada no Apêndice A, buscando facilitar a discussão dos resultados no Capítulo 4.

2 Fundamentação Teórica

Neste capítulo são abordados os conceitos e tecnologias estudados para o desenvolvimento do EHOSEP. Na seção 2.1 são apresentados os conceitos fundamentais de Computação em Grade, seguindo-se com a definição de políticas de escalonamento. Em seguida, a partir da seção 2.2, são apresentados os trabalhos relacionados, que são algoritmos de escalonamento de tarefas em grades computacionais, e em outros sistemas distribuídos relacionados a elas. Os escalonadores estudados foram propostos recentemente e consideram entre seus parâmetros o consumo de energia, critérios de justiça, ou ambos.

2.1 Definições fundamentais

Grades computacionais formam um corpo já clássico dentro de computação distribuída, sendo que o objetivo principal de sua aplicação é o compartilhamento de recursos. Nessa linha surgem dois problemas principais na sua implementação, envolvendo o acesso aos recursos e o custo desse acesso. Nas próximas páginas apresentam-se os conceitos básicos de grades, bem como os conceitos de escalonamento de tarefas (tratando o problema de acesso) e de consumo de energia (tratando parte do problema de custo).

2.1.1 Grades computacionais

Grade computacional é um tipo de sistema distribuído, que pode ser definido, de forma generalista, como recursos computacionais interligados por uma rede, e que são utilizados para resolver problemas e oferecer serviços de forma transparente. Os usuários de tais sistemas enxergam o mesmo como uma única máquina, pois os detalhes de funcionamento são omitidos, promovendo a transparência característica desses sistemas.

A interação dos usuários com os sistemas distribuídos pode ocorrer por meio da submissão de tarefas para execução, ou de outras formas como por meio de *web services*. As tarefas executadas em sistemas distribuídos podem ser de diferentes tipos, assim como podem variar os recursos computacionais utilizados, os elementos de comunicação que compõem a rede utilizada, entre outros elementos. Pode ser feita, portanto, uma classificação de tais sistemas com base nas diferentes características que podem apresentar e, entre as diferentes classes de sistemas distribuídos, encontra-se a Grade Computacional.

Grades computacionais são sistemas de *hardware* tipicamente heterogêneo, ou seja, as máquinas que compõem o sistema variam em poder computacional e, possivelmente, exercem diferentes funcionalidades. Os nós de uma grade computacional estão geralmente dispersos em alguma área, que pode ser desde o prédio de uma empresa até todo o planeta.

Com isso, as redes que interligam os nós das grades também podem variar de redes departamentais à Internet.

Em virtude de suas características, as grades computacionais são sistemas muito utilizados para promover o compartilhamento de recursos entre diferentes entidades. Nesses casos, os nós de uma grade podem estar dispersos em domínios distintos, e isso tem impacto sobre a administração do sistema, que é feita por uma camada de *software* chamada *middleware* (5).

O *middleware* deve tratar do compartilhamento de recursos, observando aspectos como autorização de acesso dos usuários, acordos firmados para a partilha de máquinas, e a qualidade de serviço esperada pelos usuários mediante os acordos firmados. Esses acordos podem ser de diversos tipos, permitindo que as grades se diferenciem umas das outras em aspectos como tempo de existência, rotas de comunicação, entre outros aspectos relacionados a negócios e geopolítica.

Foram vistas até aqui diversas características das grades computacionais, mas é necessário adotar uma definição formal para elas. Neste texto, adota-se a definição de Foster e Kesselmman (6), que consideram que um sistema distribuído deve atender a três requisitos para ser considerado uma grade:

1. Coordenação de recursos distribuídos: coordenar recursos computacionais de diferentes domínios administrativos, levando em conta segurança e aspectos políticos.
2. Padronização de interfaces: é necessário definir se as interfaces, aplicadas entre os elementos da grade, serão padronizadas ou se poderão variar. A admissão de diferentes interfaces torna mais complexo o gerenciamento do sistema.
3. Qualidade de serviço para funcionalidades não triviais: oferecer qualidade de serviço, avaliada por métricas e aspectos como tempo de resposta, disponibilidade e segurança. Tal qualidade de serviço é não trivial, pois os recursos estão dinamicamente disponíveis, os tempos de transmissão de dados podem ser altos e os caminhos para tráfego de dados podem incluir múltiplos domínios.

A variabilidade das características vistas em grades permite classificá-las (7) em alguns tipos:

- Grades Departamentais: grade definida para uso de um grupo ou departamento dentro de uma organização maior. Esse tipo de grade é de uso exclusivo ao departamento a que pertence, e não colabora com outros setores.
- Grades Empresariais: abrange os setores uma empresa, combinando recursos disponíveis em toda a entidade e permitindo a cooperação entre os departamentos da mesma.

- Grades Extra-Empresariais: agregam recursos de empresas e seus parceiros. Podem abranger clientes dessas empresas e, normalmente, utilizam redes virtuais privadas (VPN) para interconexão.
- Grades Globais: utilizam a internet para prover serviços e empregam, geralmente, recursos computacionais ociosos de seus participantes. Esses recursos podem ser emprestados ou alugados.
- Grades de Dados: são grades otimizadas para tratar acesso a dados e processamento dos mesmos. Apresentam, portanto, grande capacidade de armazenamento.
- Grades de Utilidades: são grades voltadas para o oferecimento de serviços. Por essa razão, uma de suas características principais é a alta disponibilidade de recursos e serviços.
- Grades de alto desempenho: o objetivo dessas grades é oferecer poder computacional para aplicações de alto desempenho. Para tanto, esse tipo de grade agrega grande volume de elementos de processamento, que podem variar de computadores pessoais a supercomputadores e *clusters*.
- Grades cooperativas ou colaborativas: são formadas por recursos computacionais de múltiplas entidades, que também são os usuários do sistema. Neste tipo de sistema, usuários submetem tarefas na esperança de utilizar nós de outros participantes, quando disponíveis.

Os tipos de grades não excluem um ao outro, uma grade empresarial pode ser utilizada para processamento de aplicações de alto desempenho, fazendo com que o sistema também se encaixe no tipo grade de alto desempenho. O EHOSEP procura tratar grades para compartilhamento de recursos e processamento de aplicações cuja execução está concentrada na CPU (*CPU-Bound*).

Grades cooperativas

Os usuários de grades colaborativas são os proprietários dos recursos que a compõem, e submetem tarefas na esperança de utilizar nós de outros usuários. Com essas grades é possível para um usuário obter maior desempenho sem custo financeiro. É necessário, entretanto, levar em conta que os participantes desse tipo de sistema não gostariam de ser atendidos por um poder computacional menor do que aquele que ofereceram à grade, pois esperam poder sempre utilizar seus recursos ou algo equivalente a eles.

Define-se, portanto, um critério de justiça: um usuário qualquer de uma grade cooperativa deve ser atendido com poder computacional total maior ou equivalente a

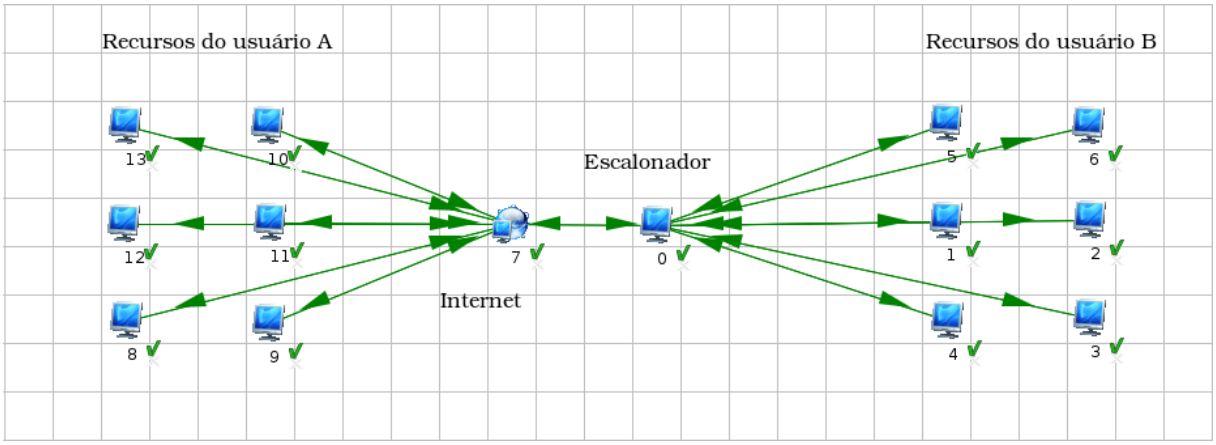

Figura 1 – Exemplo de grade computacional cooperativa.

aquele que ele ofereceu. O atendimento de tal critério é feito pelo escalonador da grade, e o algoritmo de escalonamento depende do *hardware* dos nós que compõem o sistema.

O *hardware* dos nós da grade pode ser idêntico entre eles (grade homogênea) ou pode variar (grade heterogênea). A grade pode ser de gerenciamento centralizado ou descentralizado. O gerenciamento centralizado é feito por um único nó, que faz o escalonamento e não executa tarefas dos usuários, enquanto o gerenciamento descentralizado pode é feito por múltiplos nós dos sistema. Esta estratégia de gerenciamento é aplicada no EHOSEP pois permite que ele necessite de comunicação com os outros nós do sistema em apenas dois momentos: receber tarefas e escaloná-las. Com um esquema centralizado o EHOSEP conhece o estado do sistema sem a necessidade de comunicação constante.

Caso o EHOSEP fosse um escalonador descentralizado, haveriam escalonadores locais com os quais o EHOSEP teria que trocar constantemente informações sobre o estado do sistema. Com isso, para tomar decisões de escalonamento com algum grau de certeza sobre os efeitos possíveis na grade, mantendo conhecimento confiável sobre o estado do sistema, o volume de comunicação seria grande, levando à uma limitação da escalabilidade do EHOSEP.

Um exemplo desse de grade cooperativa de gerenciamento centralizado pode ser visto na Figura 1, em que os usuários A e B compartilham 6 máquinas cada um. Estes recursos estão interligados a um escalonador, que é um nó dedicado exclusivamente à distribuição de tarefas e demais operações de gerenciamento no sistema. A conexão entre os nós e o escalonador pode ser mais direta, como em uma rede local (usuário B), ou pode ser indireta utilizando por exemplo a Internet (usuário A).

2.1.2 Escalonamento de tarefas, justiça e energia

Um dos elementos fundamentais dos sistemas computacionais é o escalonador, que define como serão executadas as tarefas do sistema, e para tanto define-se um algoritmo

de escalonamento. Nas grades computacionais os escalonadores levam em conta diversos critérios (8) para definir em que máquinas e sob quais condições as tarefas serão executadas. Tais critérios derivam de diferentes aspectos, dentre os quais cita-se alguns:

- Diminuir o tempo de entrega de tarefas no sistema;
- Obedecer critérios de justiça no uso do sistema;
- Atender requisitos de tempo, como o *deadline* de tarefas;
- Atender requisitos de tempo-real críticos em sistemas como os de controle em usinas nucleares e aeronaves;
- Otimizar o controle de acesso à memória e uso de recursos compartilhados entre tarefas;

Como pode ser observado, existem critérios ligados a desempenho, ligados à satisfação de usuários, ou mesmo ligados ao uso de recursos específicos. Desses critérios um bastante importante é o de satisfação dos usuários, pois muitas vezes os recursos disponíveis são providos por esses usuários, como em grades cooperativas. A satisfação pode ser obtida por critérios de justiça de uso, em que o uso do sistema é feito de forma mais harmônica entre os vários usuários.

O critério de justiça de uso é bastante antigo, datando pelo menos de 1984 em sistemas UNIX (9) ou antes disso se forem considerados sistemas de tempo compartilhado. Embora muitos dos algoritmos baseados em justiça considerem o histórico de uso como critério de justiça, alguns fazem uso de outros parâmetros, como o de propriedade de recursos por exemplo. A ideia por trás de usar a propriedade como critério de justiça é interessante em grades cooperativas, uma vez que se usuários forem estimulados a compartilhar recursos então todos ganham de um modo ou outro.

O trabalho aqui apresentado se baseia em uma política de escalonamento baseada em justiça de uso pelo conceito de propriedade (2), descrita a seguir em 2.4. Com isso se estimula o compartilhamento pois num dado instante um usuário terá disponível tanto ou mais poder computacional do que é proprietário. Entretanto, é preciso considerar que compartilhar recursos pode elevar os custos com consumo de energia, pois tarefas de outros participantes do sistema poder ser executadas nas máquinas que o usuário integrou ao sistema.

Como o escalonamento define em que máquinas as tarefas serão executadas, ele também define quais máquinas ficarão ativas no sistema e, de maneira direta ou indireta, como serão os caminhos para troca de dados entre as tarefas durante a execução. Além disso, a eficiência energética de cada nó da grade pode ser diferente, o que faz com que o

consumo de energia do sistema passe a depender de como ocorre a alocação das tarefas entre as máquinas. Isso ocorre pela diferença entre máquinas, tráfego de dados entre as tarefas e, em menor grau, pela diferença entre componentes de armazenamento (memória e discos). Mesmo que um escalonador desconsidere o consumo de energia ao tomar as decisões de alocação, ele ainda exercerá grande impacto no consumo dos sistemas.

Os escalonadores podem levar em conta o consumo de energia como parâmetro de decisão, e trabalhar, por exemplo, para otimizar rotas de comunicação, considerar a geração de calor, e até aplicar técnicas que reduzem ativamente o consumo. Deve ficar claro também que apenas usar consumo de energia como critério de escalonamento pode prejudicar outros aspectos como tempo de execução das tarefas, cumprimento de *deadlines* de tarefas, entre outros. Assim, é importante também levar em consideração aspectos como desempenho das aplicações ou critérios de justiça de uso.

Nesse cenário fica claro que o escalonamento de tarefas afeta tanto a justiça no uso dos recursos do sistema como o consumo de energia, e que é interessante que sejam considerados ambos os aspectos dos sistemas de forma concomitante. A maioria dos algoritmos de escalonamento considera apenas um dos aspectos, conforme pode ser visto nas seções 2.2, cuidando apenas de consumo de energia, e 2.3, cuidando de justiça de uso. Posteriormente serão tratados algoritmos que combinam energia e justiça na Seção 2.5.

2.2 Algoritmos de escalonamento baseados em consumo de energia

Os algoritmos de escalonamento baseados em consumo de energia usam técnicas diversas para sua redução. Nessa categoria entram algoritmos atuando no nível eletrônico, no nível de componentes, ou mesmo no nível da arquitetura da grade, como será visto a seguir.

2.2.1 Soluções baseads em DVFS

Uma das técnicas mais utilizadas para tratar o consumo de energia é o DVFS (*Dynamic Voltage and Frequency Scaling*), que altera a tensão da corrente elétrica que alimenta o processador e a frequência do relógio, permitindo reduzi-los para poupar energia. Propostas utilizando DVFS atuam na redução de energia em nível eletrônico, sendo que os algoritmos nesta linha trabalham aspectos distintos dos circuitos.

No escalonador HGS-Sched (*Hierarchic Genetic Scheduler*) (10) o DVFS é utilizado para colocar máquinas inativas em um estado de potência mínima, enquanto para máquinas ativas o DVFS é aplicado para consumir o mínimo de energia possível enquanto é garantido o cumprimento do *deadline*. No escalonador RMEC (*Reliability Maximization with Energy Constraint*) (11) o DVFS também é utilizado para reduzir o consumo de energia na

execução das tarefas, mas seu uso é limitado pelas regras de precedência entre as tarefas que o escalonador trata.

Já o algoritmo ESTS (*Energy-aware Stochastic Task Scheduling*) (12) combina o uso de ferramentas estatísticas com DVFS para maximizar a chance de respeitar o *deadline* de tarefas. As tarefas que ele trata possuem um orçamento que a execução de cada uma pode consumir. O algoritmo de escalonamento SEATS (*Smart Energy-Aware Task Scheduling*) (13) tenta reduzir o consumo adiantando o desligamento de máquinas hospedando máquinas virtuais, acelarando sua excecução por meio do DVFS. O SEATS distribui o processamento extra entre as tarefas de uma VM de acordo com uma política definida pelo administrador do sistema.

Em conjunto com as técnicas dos algoritmos genéticos, o algoritmo OL-PICEA-g (*Opposition based learning preference-inspired co-evolutionary algorithm - goal vector*) (14) aplica DVFS para maximizar o uso de energias renováveis, e de fornecimento variável ao longo do dia. Neste caso o DVFS é aplicado para reduzir consumo em períodos em que a energia utilizada é não renovável, e sua aplicação está sujeita ao cumprimento de *deadlines* de tarefas.

Uma outra abordagem aparece no algoritmo CloudFreq (15), que considera o conflito entre o tempo para concluir todas as tarefas e consumo, e busca obter uma redução do consumo de energia com DVFS sem prejudicar demasiadamente o tempo total de execução do conjunto de tarefas escalonado. Já o escalonador EATS-FFD (*Energy Aware Task Scheduling - First Fit Decreasing*) (16) aplica DFVS em conjunto com técnicas de reutilização de máquinas virtuais em sistemas de computação em nuvem, limitando o DVFS de forma que sejam atendidos os *deadlines* das tarefas.

2.2.2 Soluções baseadas no tratamento da infraestrutura

A infraestrutura dos sistemas computacionais tem grande influência sobre o consumo deles. Componentes de arrefecimento, as fontes de energia e baterias, controle de gabinetes em *datacenters* e instrumentos de monitoramento do estado do sistema são alguns dos elementos cuja atividade e disponibilidade varia com a ocupação do sistema, afetando o consumo de energia total do mesmo e seu desempenho. Com isso, muitos escalonadores passaram a tratar tais elementos em suas decisões de escalonamento.

O algoritmo de escalonamento proposto por Tchernykh (17) procura reduzir o consumo desligando máquinas da grade. Isso é feito desligando máquinas que ficam inativas por um período de tempo determinado pelo administrador do sistema. Além disso também é feita a replicação de tarefas, mantendo um balanceamento entre o número de réplicas e o aumento no consumo de energia gerado com elas. O escalonador *MinD+ED* (*Minimum Dependencies Energy-efficient DAG*) (18) também procura desligar nós ociosos, mas aplica

técnicas de minimização de dependências em tarefas do tipo DAG (*Directed Acyclic Graph*).

O escalonador EARH (*Energy-Aware Rolling-Horizon*) (19) procura agrupar máquinas virtuais em servidores de forma a reduzir a quantidade deles em atividade. O escalonador procura também reduzir ao mínimo o número de máquinas virtuais que atendem a cada tarefa, sem que o *deadline* delas seja desrespeitado. Já o escalonador proposto por Comito (20) emprega agrupamento e migração de tarefas em grades de dispositivos sem fio, como *smartphones*, de maneira a maximizar o tempo de vida do sistema levando em conta a carga das baterias dos nós, e quantidade deles em atividade.

O escalonador OHTA (*Optimal Heterogeneous Task Assignment*) (21) também escalona tarefas em redes de dispositivos móveis. Ele procura maximizar o tempo de atividade do sistema transferindo tarefas para sistemas de nuvem externos, de forma a considerar tanto o consumo da grade móvel quanto do sistema externo. Já o sistema de computação em nuvem *Green Control Centre* (22) procura escalonar tarefas em horários de energia mais barata e proveniente de fontes renováveis, sem o uso de DVFS. O uso de energias renováveis também é explorado nos escalonadores dos sistemas EpoCloud (23) e GreenMR (24).

2.2.3 Soluções baseadas na localização e agrupamento de dados

A transferência de dados entre tarefas apresenta consumo de energia e, caso seja postergada, pode exigir a paralisação ou atraso na execução das tarefas envolvidas. Isso pode levar a um aumento no consumo de energia para executá-las. Zhao (25) propõe um algoritmo de escalonamento que agrupa as tarefas com base na correlação existente entre os dados necessários para elas. Tarefas que utilizam os mesmos dados são alocadas para as máquinas que os contém, reduzindo consumo de energia com transferência e replicação de dados.

Maqsood (26) desenvolveu escalonadores que fazem o agrupamento de dados em memória cache de processadores *multicore*, com núcleos interconectados via *Network-on-Chip* (NoC). O agrupamento ocorre de forma que os dados estejam em níveis de cache próximos dos núcleos em que as tarefas que deles necessitam estiverem executando. Tal agrupamento reduz o consumo de energia com intercomunicação entre núcleos, melhorando o tempo de acesso aos dados. Isso reduz o tempo de execução das tarefas e, consequentemente o consumo de energia com execução também. Outro algoritmo que considera o posicionamento dos dados em processadores multicore NoC é proposto por Chatterjee (27).

2.2.4 Soluções com tratatamento da heterogeneidade de recursos

Em sistemas de *hardware* heterogêneo o escalonador pode considerar a variação no consumo e no poder computacional dos nós. Isso pode ser feito enviando tarefas para nós que executem consumindo menos energia, ou que precisem de menos tempo para concluir a tarefa. Com a rápida evolução do *hardware* e dos meios de comunicação em rede esse tipo de abordagem tem ocorrido com frequência nos escalonadores recentes.

O escalonador proposto por Dorronsoro (28) atua com tarefas do tipo DAG em sistemas de múltiplos *clusters*/*grids*, enviando tarefas para nós em que o consumo de energia previsto é menor, em virtude de *hardware* mais eficiente energeticamente. O algoritmo considera também máquinas que promovam redução significativa no tempo de execução de todo o conjunto de tarefas (*makespan*). De forma similar Tarplee, Maciejewski e Siegel (29) propõem um escalonador que procura balancear *makespan* e consumo de energia em sistemas de *hardware* heterogêneo, considerando como parâmetro de decisão o lucro previsto do provedor de recursos e serviços.

Já os algoritmos EMRSA-I e EMRSA-II (*energy-aware MapReduce scheduling algorithms I e II*) (30) tentam reduzir o consumo de energia pela alocação de tarefas MapReduce de maior tamanho para recursos físicos mais eficientes energeticamente. Li (31) propõe escalonadores que têm foco no número de máquinas virtuais alocadas para tarefas e no consumo que varia nos recursos físicos heterogêneos. Busca-se assim escolher *hosts* de menor consumo e empregar técnicas de negociação de SLA para flexibilizar *deadlines*, quando necessário, para economizar energia.

O escalonador ENRG-MaxMin (*Energy MaxMin*) (32) verifica quais recursos computacionais podem executar as tarefas honrando o *deadline* de cada uma delas. Com tal restrição ele envia as tarefas para o nó que, entre tais recursos, apresentar menor consumo de energia. O algoritmo TaPRA (*Task Placement and Resource Allocation*) (33), e suas variações, procura escalonar *jobs* (conjuntos de tarefas) para máquinas virtuais. Cada *job* tem um orçamento de energia para sua execução, e as VMs escolhidas são aquelas que estiverem hospedadas no servidor que consumir menos energia para executar as tarefas.

2.2.5 Soluções baseadas em algoritmos bioinspirados

Técnicas inspiradas em processos biológicos e comportamento de animais também são empregadas em algoritmos de escalonamento e alocação de tarefas. Assim também se encontram propostas usando algoritmos bioinspirados para tratar escalonamentos tendo consumo de energia como política. Além do PICEA-g (visto em 2.2.1), o escalonador proposto por Iturriaga, Dorronsoro e Nesmachnow (34) emprega algoritmos evolutivos para tomar as decisões de escalonamento. Já o algoritmo EAMOCA (*Multi objective Chiropteran Algorithm*) (35) baseia-se no comportamento dos morcegos, enquanto o escalonador CLPS-

GA (*Case library and Pareto solution-based hybrid Genetic Algorithm*) (36) emprega algoritmo genético.

2.3 Algoritmos de escalonamento baseados em critérios de justiça

Como visto em 2.1.2, critérios de justiça em escalonamento podem ser vistos a partir de conceitos diversos. Assim apresenta-se aqui algumas dessas visões de justiça, com destaque ao critério a ser usado neste trabalho, que é o de justiça por propriedade de recursos.

A visão mais básica de justiça é a apresentada por Amar, Kay e Lauder (37, 38), que tratam o histórico de uso como sendo a base de justiça. Nessas abordagens usuários que tenham usado pouco a grade num passado recente teriam prioridade no uso atual. Uma proposta similar é feita por Andrade (39), que busca priorizar mais aos usuários que tenham favorecido o uso de outros usuários recentemente.

Critérios distintos de justiça aparecem em algoritmos como o CGPA (*Cost-Greedy Price-Adjusting*) (40), que considera como justiça o lucro obtido com a venda de tempo de uso. Nele procura-se garantir que, em um sistema formado com recursos de diferentes provedores para atender a usuários externos, os provedores tenham chances iguais de vender processamento e, portanto, lucrar com isso. Já o escalonador proposto por Xiao (41) utiliza Teoria dos Jogos para escalonar tarefas em grades cooperativas, de forma a garantir que o prejuízo no tempo de execução das tarefas locais de cada participante, em virtude da execução de tarefas de outros provedores, seja igual para todos os provedores de recursos na grade. Já o escalonador Chronos (42) pode incorporar, dentre outras estratégias, a divisão igualitária do tempo de CPU entre tarefas MapReduce.

De forma similar ao Chronos, a divisão igualitária de recursos também pode ser utilizada para evitar espera indefinida de tarefas por recursos (*starvation*) em sistemas de computação em nuvem (43, 44). Já o gerenciador Borg (45), utilizado pela Google, emprega um esquema de múltiplas filas de prioridade com Round-Robin entre tarefas de uma mesma fila, promovendo justiça e evitando *starvation*. Um esquema semelhante é utilizado no algoritmo proposto por Saxena, Chauhan e Kait (46).

O algoritmo T_DMHSV (*tradeoff-based fairness-based dynamic multiple heterogeneous selection value*) (47) escalona tarefas DAG de múltiplas prioridades, cada uma com seu *deadline*. O objetivo do T_DMHSV é garantir o cumprimento do *deadline* de tarefas de alta prioridade, mas sendo justo com tarefas menos prioritárias ao garantir que todas tenham chance de executar.

2.4 Trabalhos prévios: algoritmos OSEP e HOSEP

O problema atacado com o algoritmo CGPA (40) pode ser resumido em "como estimular um dono de recursos a compartilhá-los numa grade". Falavinha desenvolveu o algoritmo OSEP (*Owner-Share Enforcement Policy*) (2). Como a proposta apresentada nesta dissertação está estruturada a partir do OSEP é interessante fazer aqui uma descrição mais completa do OSEP e de sua variação para ambientes heterogêneos, o HOSEP.

Em linhas gerais o comportamento desses algoritmos é o seguinte:

- É executado em intervalos regulares definidos pelo administrador do sistema.
- Divide-se em duas partes: algoritmo de atualização e algoritmo de decisão.
- O algoritmo de atualização obtém informações sobre tarefas em execução, demanda não atendida dos usuários e máquinas que atendem cada um destes.
- O algoritmo de decisão é chamado pelo algoritmo de atualização, e é responsável por alocar tarefas que aguardam na fila para recursos livres, assim como fazer preempção de recursos quando necessário, e possível, para atender o critério de justiça.
- Os usuários que num dado momento estejam sendo atendidos por mais recursos que forneceram são os candidatos a ceder recursos para aqueles que não atingiram suas quotas.

Como o OSEP trata apenas o número de recursos que atende a cada usuário, ele não cumpre seu objetivo em grades de *hardware* heterogêneo. Nessas grades o poder computacional varia entre nós distintos e, portanto, não é possível atender ao critério de justiça apenas levando em conta o número de máquinas que atende cada usuário.

Forte (3) estendeu o algoritmo OSEP, adaptando-o para sistemas heterogêneos com a introdução do algoritmo HOSEP (*Heterogeneous OSEP*). O HOSEP tem o mesmo comportamento básico do OSEP, ou seja, procura tirar recursos de usuários com excesso para atender aqueles que ainda não são atendidos pelas suas quotas no sistema, porém agora levando em consideração o poder computacional individual de cada recurso.

O algoritmo OSEP calcula excesso/falta de recursos subtraindo o tamanho da quota de um usuário do número de máquinas usadas por ele. O HOSEP, por outro lado, deve levar em conta o poder computacional dos recursos e, para tanto, utiliza uma métrica chamada Diferencial de Poder (DP). Dado um participante i da grade, atendido por um poder computacional correspondente a $Alocado_i$ MFLOPS e fornecedor de uma quota cujo

desempenho total é de $Quota_i$ MFLOPS, calcula-se o diferencial de poder desse usuário por meio da equação 2.1.

$$DP_i = \frac{Alocado_i - Quota_i}{Quota_i} \tag{2.1}$$

Usuários com excesso de poder computacional já alocado para si apresentarão diferencial de poder maior que zero e, quanto maior o valor, maior o excedente. Se houver falta de poder computacional o DP será negativo e, quanto mais negativo, maior será a defasagem. Além do Diferencial de Poder, calcula-se também uma métrica chamada Poder Remanescente Esperado (PRE), que é calculado para decidir se uma preempção deve ser feita de forma que, dada uma máquina y que está executando uma tarefa do usuário z, e que pode sofrer preempção para atender a uma tarefa do usuário w, calcula-se o valor de PRE para ambos os usuários com as equações 2.2 e 2.3.

$$PRE\_Preemp_{z,y} = \frac{Alocado_z - Quota_z - Poder_y}{Quota_z} \tag{2.2}$$

$$PRE\_Aloc_{w,y} = \frac{Alocado_w - Quota_w + Poder_y}{Quota_w} \tag{2.3}$$

O HOSEP escalona tarefas individuais, e independentes entre si, a máquinas do sistema que executam apenas uma delas por vez. O HOSEP está especificado no algoritmo 1, e procura garantir que cada participante de uma grade cooperativa heterogênea seja atendido por um poder computacional maior ou igual ao que ele ofereceu à grade.

Inicialmente, o HOSEP define na linha 2 que o usuário atendido deve ser aquele que apresentar o menor diferencial de poder. Em seguida, nas linhas 3 a 5, procura-se por nós disponíveis e, se houver, é escolhido aquele que tenha o maior poder computacional.

Caso não haja nós ociosos na grade e o usuário em atendimento apresenta defasagem de poder computacional ($DP_i < 0$), o HOSEP tenta fazer preempção, começando pela busca do usuário com maior excesso de poder computacional, em relação à sua quota. Esse usuário apresenta o DP de maior valor (linha 7). A preempção só ocorre se uma de duas condições forem satisfeitas (linha 8):

- O Poder Remanescente Esperado do usuário que perderia o recurso é maior que o DP do usuário que receberia o recurso;
- O Poder Remanescente Esperado do usuário que perderia o recurso é positivo;

Caso sejam encontradas múltiplas máquinas que atendam às condições acima, é escolhida aquela que apresentar o menor poder computacional a fim de minimizar o impacto sobre o diferencial de poder do usuário que perde recurso.

```
1  enquanto não houver executado todas as tarefas faça
2      se ∃ usuário u_i | DP_i = min(DP_1, ..., DP_m) então
3          se ∃ conjunto de máquinas livres (M_f, ..., M_g) então
4              Escolhe máquina M_h | Desempenho_h =
                max(Desempenho_f, ..., Desempenho_g);
5              Reserva M_h para a menor tarefa do usuário u_i que estiver na fila;
6          senão
7              se DP_i < 0 ∧ ∃ usuário u_j | DP_j = max(DP_1, ..., DP_m) ∧ DP_j > 0
                então
8                  se ∃ tarefa k de u_j executada em y | [(PRE_Preemp_{j,y} ≥
                    PRE_Aloc_{i,y}) ∨ (PRE_Preemp_{j,y} ≥ 0)] então
9                      Realiza preeempção da tarefa k do usuário u_j;
10                     Reserva recurso para a menor tarefa, da fila, do usuário u_i;
11                     Atualiza dados dos usuário e das tarefas;
12                 fim
13             fim
14         fim
15     fim
16 fim
```

Algoritmo 1: HOSEP

O algoritmo não força a execução de tarefas nos recursos do usuário que as submeteu, nem faz rearranjos completos da carga de trabalho para atender, com exatidão, à quota de cada usuário. Este comportamento, combinado com as condições para preempção, leva à possibilidade da defasagens de uso, ainda que pequenas. Essas defasagens estão previstas no critério de justiça, ligeiramente relaxado em relação ao critério do OSEP, e detalhado a seguir.

Para o algoritmo HOSEP, um usuário x está sendo atendido de forma justa se uma das seguintes condições for atendida:

i) $DP_x \geq 0$.

ii) $DP_x < 0 \Rightarrow \forall$ *usuário* $i \wedge$ *máquina* $j \mid DP_i \geq 0 \wedge j$ *está alocado para* $i \mid [(PRE\_Preemp_{i,j} < PRE\_Aloc_{x,j}) \vee (PRE\_Aloc_{x,j} < 0)]$

A primeira condição diz que o usuário está sendo atendido com justiça se utilizar mais poder computacional do que disponibilizou. Já a segunda condição diz que um usuário está sendo atendido com justiça, mesmo recebendo menos volume de processamento do que disponibilizou, quando, ao receber mais levará a uma situação de injustiça ainda maior para outro usuário.

A segunda condição indica que o critério de justiça do OSEP, que era exato, teve que ser relaxado para o HOSEP em virtude das diferenças no desempenho dos nós do

sistema. Esse relaxamento não prejudica o cumprimento dos objetivos, e evita desperdício de processamento ao reduzir o número de preempções necessárias. Os resultados dos testes feitos com o EHOSEP (3) mostram que o algoritmo HOSEP garante o cumprimento do critério de justiça para todos os usuários.

2.5 Algoritmos de escalonamento baseados em energia e justiça

O tratamento do consumo de energia vem recebendo significativa atenção dos pesquisadores. Já os critérios de justiça receberam cuidados em menor intensidade, enquanto são poucos os escalonadores que consideram tanto energia como justiça. Serão vistos a seguir os escalonadores recentes que tratam simultaneamente energia e justiça.

2.5.1 *Power Credit based Fair scheduler*

Trata-se de um *framework* para escalonamento de máquinas virtuais em sistemas de computação em nuvem. Os recursos computacionais são máquinas com processadores *multicore* de núcleos homogêneos. O escalonador foi batizado PCFS (48), e tem por objetivo não apenas partilhar o tempo de CPU entre as VMs, mas também reduzir a energia consumida para hospedá-las e executar suas tarefas.

Cada máquina virtual recebe créditos de CPU e créditos de energia, que representam respectivamente o tempo de CPU e o consumo de energia que a VM pode ter. A definição da quantidade de créditos das VMs e, portanto, o tempo de CPU e a energia consumida por elas, são feitas de forma a atender aos requisitos de cada VM.

A fim de reduzir o consumo de energia, o PCFS utiliza DVFS nos núcleos do processador, criando grupos de núcleos. Cada grupo é chamado *Power Zone*, e seus núcleos operam com o mesmo par frequência-tensão. Os *Power Zones* são definidos de forma a atender, com o menor consumo de energia possível, os requisitos das máquinas virtuais. Além da frequência e da tensão, há outros fatores que determinam o agrupamento dos núcleos de processamento; que são:

- Topologia: é levado em conta o compartilhamento de memória cache entre os núcleos. Procura-se agrupá-los de forma que possam trocar dados através dos bancos de cache a que estão diretamente ligados;
- Distribuição de calor: procura-se balancear a geração de calor entre os núcleos. Núcleos que atingem determinado teto de temperatura tem sua frequência reduzida, sendo substituídos por núcleos mais frios, que estavam em grupos de consumo mais baixo;

- Histórico de ocupação: é mantido um histórico da ocupação de cada núcleo e, se um núcleo apresentar ocupação inferior a 20% durante 8 medições seguidas, ele poderá ser desligado ou transferido para grupos de frequência mínima de operação;

As VMs cujos requisitos são atendidos por determinado *Power Zone*, e que ainda tem créditos, compartilham o tempo de uso dos núcleos dele de maneira igualitária. Quando os créditos de uma VM acabam ela deixa de utilizar a CPU e o tempo de execução que ela ocupava é dividido igualmente entre as VMs que ainda tem créditos. A reposição de créditos ocorre em intervalos de 30ms.

O PCFS é interessante para reduzir o consumo de energia em sistemas de computação em nuvem, pois faz essa redução respeitando os requisitos de cada máquina virtual e, portanto, não prejudica o desempenho das aplicações. Entretanto, a submissão aos requisitos das VMs faz com que a redução do consumo fique dependente da carga de trabalho e, com isso, a atuação do algoritmo pode ser limitada.

2.5.2 *Energy Fair Share*

O algoritmo *Energy Fair Share* (EFS) (49) foi proposto para escalonamento de tarefas em *datacenters* e tem por objetivo incentivar os usuários a consumir menos energia, seja reduzindo o tempo de uso de CPU nas suas aplicações, seja otimizando suas aplicações para que haja maior eficiência energética.

O EFS baseia-se no algoritmo *Fair Share*, que foi desenvolvido para dividir igualmente o tempo de CPU entre usuários. No *Fair Share*, processos que tiveram pouco tempo de CPU recebem alta prioridade, enquanto aqueles que ocuparam a CPU por mais tempo passam a ser pouco prioritários.

Em vez de considerar o tempo de CPU, o EFS leva em conta a energia consumida pelas tarefas ao longo do tempo. O EFS dá alta prioridade de execução para tarefas que consumiram pouca energia ao longo do tempo, enquanto tarefas que consumiram muita energia recebem baixa prioridade, ou podem ser forçadas a executar apenas quando o sistema estiver ocioso.

O consumo de energia das tarefas é dado pelo somatório ponderado do consumo medido periodicamente. Cada medição é multiplicada por um peso e, conforme o tempo avança, as medições mais antigas vão recebendo pesos cada vez menores, a fim de que as medições mais recentes tenham mais influência sobre a definição de prioridade.

O EFS é um algoritmo simples, e a divisão do consumo de energia, considerando-o como recurso principal em vez do tempo de CPU é muito interessante no cenário atual dos *datacenters*, pois os custos relacionados a esse consumo tornaram-se grandes o suficiente para tratar-se energia como recurso fundamental. Por outro lado, manter o histórico de uso

do sistema em *datacenters* de grande porte pode ser proibitivo, e o algoritmo não procura reduzir o consumo de energia com outras técnicas como o DVFS, o que seria interessante nos sistemas tratados por ele.

2.5.3 *Energy-Credit Scheduler*

Sistemas de computação em nuvem cobram de seus usuários com base nos recursos utilizados ao longo do tempo. Dessa forma, usuários pagam por tempo de CPU de suas VMs, espaço ocupado em disco, quantidade de memória utilizada, etc.

É possível, por outro lado, considerar o consumo de energia como um recurso, como foi feito no *Energy Fair Share*. O algoritmo *Energy-Credit Scheduler* (50) escalona máquinas virtuais utilizando um sistema de créditos de energia, de maneira idêntica ao algoritmo anterior. Mas, nesse caso, os créditos representam o consumo de energia contratado pelos usuários para suas VMs.

O escalonador atribui créditos de energia em intervalos chamados períodos fiscais. No início de cada período fiscal as máquinas virtuais recebem créditos, e o tempo de CPU é igualmente compartilhado entre as VMs que ainda tem créditos. Quando os créditos de uma VM acabam, ela sai da fila do escalonador e só retorna no próximo período fiscal, quando receber novos créditos.

O *Energy-Credit Scheduler* é semelhante ao EFS, mas não divide o consumo de energia de forma igualitária, e sim de acordo com o consumo de energia contratado. Com isso não é necessário manter histórico de uso, mas apenas verificar o consumo de cada máquina virtual ao longo dos períodos fiscais. A ausência de histórico torna o *Energy-Credit Scheduler* mais interessante que o EFS, embora ele não tente reduzir o consumo.

2.5.4 *Energy-Aware Multi-Organization Scheduling Problem* (MOSP-Energy)

Trata-se de um escalonador de tarefas em plataformas computacionais cooperativas, que englobam as grades cooperativas e os sistemas de computação em nuvem comunitários. O escalonador foi batizado de MOSP-energy (51), sendo o único dos algoritmos encontrados que trata consumo de energia no mesmo tipo de sistema em que o EHOSEP foi projetado para trabalhar.

O *MOSP-energy* tem por objetivo reduzir o consumo de energia nos sistemas considerados, sem permitir que os usuários participantes tenham custos operacionais maiores com energia, em comparação com aqueles que teriam se não participassem do sistema. As tarefas escalonadas apresentam *deadline*, e o consumo das máquinas do sistema é variável e dependente da velocidade de processamento, que é controlada para reduzir o consumo de energia.

O algoritmo trabalha principalmente com migração de tarefas e redução da velocidade de processamento. Estes dois procedimentos são restritos ao não aumento dos custos das tarefas de cada participante, e ao cumprimento dos *dealines*. Para tratar estas restrições o escalonador busca agrupamentos de tarefas que, nos recursos em que serão executados, contenham a maior quantidade de tarefas possível, consumam a menor quantidade de energia possível e respeitem o *deadline* de todas as tarefas do agrupamento.

2.5.5 *Max-Max Utility-Per-Energy Consumption (Max-Max UPE)*

O algoritmo Max-Max UPE (52) é utilizado para escalonar tarefas independentes entre si, em um ambiente de *hardware* heterogêneo utilizando DVFS. A cada tarefa é atribuído um valor de utilidade, que é uma métrica para quantificar a utilidade de se executar a tarefa. O valor de utilidade cai conforme cresce o tempo previsto para completar a execução da tarefa.

No sistema considerado há um limite de consumo anual, e o objetivo do escalonador é maximizar a utilidade obtida no sistema com a energia disponível no ano. A utilidade do sistema é o somatório das utilidades das tarefas executadas. Para atingir seu objetivo o escalonador calcula o limite diário de consumo, dividindo o consumo restante para o ano pelo número de dias restantes no mesmo. Além disso o objetivo é simplificado de forma que passe a ser a maximização da utilidade obtida no dia, respeitando o limite de consumo total diário.

Durante o dia a energia restante para execução nele é dividida entre as tarefas de forma aproximadamente igualitária, permitindo-se que algumas tarefas consumam mais energia que outras. Isso é feito por meio de um filtro de energia implementado no escalonador. O filtro se comporta de acordo com parâmetros definidos pelo administrador para permitir ou não tarefas com maior consumo que as demais.

O escalonador atua em dois estágios. No primeiro, o escalonador verifica as possíveis combinações de tarefa, máquina e nível de DVFS para a tarefa em alocação. No segundo estágio é escolhida a combinação que levar ao maior valor de utilidade/consumo na execução da tarefa em particular.

2.6 Considerações finais

A revisão bibliográfica apresentada nas seções anteriores permite constatar diferentes abordagens para políticas de escalonamento em grades baseadas em consumo de energia. Do mesmo modo o conceito de escalonamento baseado em justiça apresenta diversas propostas distintas. Com isso se propõe neste trabalho uma política de escalonamento compondo consumo de energia e justiça baseada no conceito de propriedade. O algoritmo

Tabela 1 – Comparação dos algoritmos que consideram energia e justiça.

Algoritmo	Critério de justiça	Varia velocidade da CPU (DVFS)	Otimiza consumo com infraestrutura	Prioriza usuários que consomem menos e/ou tem nós eficientes	Escalona em grade ou nuvem cooperativa	Tenta reduzir o consumo total do sistema
Power Credit based Fair scheduler	Desempenho e Energia	✔	✔			✔
Energy Fair Share	Energia			✔		
Energy-Credit Scheduler	Desempenho e Energia			✔		
MOSP-energy	Desempenho e Energia	✔			✔	✔
Max-Max UPE	Desempenho e Energia	✔				
EHOSEP	Desempenho e Energia			✔	✔	✔

Fonte: do autor

proposto, EHOSEP, busca a redução no consumo de energia ao favorecer usuários que ou consumam menos energia ou forneçam recursos computacional e energeticamente eficientes.

Como o EHOSEP considera tanto energia quanto justiça como critérios de escalonamento, apresenta-se na Tabela 1 a comparação da proposta deste trabalho com escalonadores semelhantes a ele. Nela pode ser observado que a técnica mais usada para a redução no consumo de energia é a aplicação de DVFS, sendo que apenas um ainda busca a otimização do consumo com a infraestrutura global do sistema.

Os demais algoritmos buscam a redução por meio de priorização dos usuários com menor consumo. Nesta categoria se enquadra o EHOSEP, que também é, junto com o MOSP-energy um dos que atuam em grades cooperativas. Deve ser observado que nessas grades a redução no consumo de energia é importante para quem as usam pois esses usuários são também os provedores dos recursos.

Mesmo trabalhando com sistemas idênticos aos tratados pelo EHOSEP, o *MOSP-energy* difere muito dele, pois apresenta um critério de justiça baseado no custo operacional de cada usuário. Com isso as técnicas usadas tratam a velocidade de processamento dos nós, tendo como alvo tarefas com *deadline*. A variação de velocidade por meio do DVFS, embora interessante, não é aplicada no EHOSEP pois causaria grande aumento na complexidade

do tratamento de justiça de propriedade. No EHOSEP se obtém a justiça de propriedade pela garantia de uso de poder computacional equivalente ao das máquinas compartilhadas, o que seria difícil com a variação no poder de processamento gerada pelo DVFS.

3 Algoritmo EHOSEP

Neste capítulo é apresentado o algoritmo de escalonamento proposto. Serão definidos os objetivos do escalonador, sua formulação e os critérios por ele utilizados. A formalização do algoritmo é seguida com sua validação algébrica, em especial seu critério de parada e atendimento dos objetivos de justiça e consumo de energia.

3.1 Objetivos do EHOSEP

Do ponto de vista prático o consumo de energia pode ser minimizado de duas formas: 1-equipamentos energeticamente eficientes ou 2-redução do consumo ou demanda. A primeira forma implica em equipamentos com melhor razão MFLOP/Joule enquanto a segunda em reduzir a demanda por processamento.

O EHOSEP foi projetado para favorecer usuários cujos recursos contribuam significativamente para o poder computacional geral da grade, e que sejam eficientes do ponto de vista energético. Além disso, usuários que restrinjam o consumo de energia, por meio de um limite de potência instantânea, também são favorecidos. Estimular a oferta de recursos eficientes e de poder computacional significativo promove o crescimento do poder computacional do sistema, sem provocar grande aumento no consumo dele. Ou seja, o poder computacional da grade aumenta mais do que o consumo, levando a um sistema mais eficiente energeticamente. Ter usuários limitando a potência instantânea com que são atendidos provoca naturalmente uma redução do consumo total no sistema. Conforme será visto mais adiante, o EHOSEP procura alocar tarefas para as máquinas de menor potência.

Isso leva à definição de um conjunto de premissas para atendimento do EHOSEP, que são:

1. A busca por justiça de propriedade deve ser prioritária;
2. Essa busca deve ocorrer condicionada aos limites de potência definidos pelo usuário;
3. A eficiência energética da grade deve ser buscada pela priorização de máquinas de melhor relação MFLOP/Watt.

3.2 Escalonador EHOSEP

A definição do EHOSEP parte de um algoritmo para a sua implementação, definição dos parâmetros operacionais e de métricas para a sua avaliação/validação. Nesta seção se

apresentam esses detalhes, ficando a validação teórica de seu funcionamento para a parte final deste capítulo.

3.2.1 Comportamento básico do EHOSEP

Sendo um metaescalonador, a ativação do EHOSEP ocorre periodicamente, de forma a não sobrecarregar o sistema. Em cada ativação, que pode ocorrer a cada 30 segundos, o algoritmo busca atender tarefas de todos os usuários que tenham ainda disponibilidade de carga e potência por meio do algoritmo 2:

```
1  enquanto houver usuários em condições de atendimento faça
2  |  Ordene os usuários de acordo com a necessidade de poder de processamento;
3  |  Partindo do usuário com maior falta de poder computacional, até o usuário com
   |   menor falta ou maior excesso de recursos, execute o laço a seguir;
4  |  enquanto não houver tentado atender a todos os usuários faça
5  |  |  Tente alocar máquina livre para usuário em atendimento;
6  |  |  se não foi possível atender com máquina livre então
7  |  |  |  Tente fazer preempção de máquinas de outros usuários;
8  |  |  fim
9  |  |  se houve atendimento então
10 |  |  |  saia do laço de repetição interno;
11 |  |  fim
12 |  fim
13 fim
```

Algoritmo 2: EHOSEP simplificado

Para executar suas funções, o EHOSEP utiliza alguns parâmetros, descritos a seguir. Estes parâmetros incluem uma variável de controle (AT) para indicar se um usuário foi atendido, e parâmetros que descrevem as quotas de recursos dos usuários, o poder computacional das máquinas, a potência das mesmas, a tarefas submetidas pelos usuários, entre outros.

O EHOSEP também utiliza métricas de decisão indicando a falta ou excesso de recursos com que os usuários são atendidos, quais usuários devem ser os primeiros candidatos a perder máquinas em procedimentos de preempção e como os usuários contribuem para a grade. Tanto os parâmetros como as métricas utilizadas são apresentados a seguir, e posteriormente é apresentado o comportamento detalhado do EHOSEP com os algoritmos 3, 4 e 5.

3.2.2 Parâmetros de projeto

O algoritmo EHOSEP tem sua formulação baseada em alguns parâmetros, como quem compartilha a grade e de que forma, além de aspectos relativos ao que é executado. Esses parâmetros são apresentados a seguir:

Lista de Usuários

No EHOSEP os usuários são organizados na forma de uma lista, expressa no conjunto $U = \{u_1, u_2, ..., u_N\}$ contendo os N usuários do sistema. Cada usuário u_i apresenta características distintas quanto às máquinas que oferece e tarefas que executa, entre outros parâmetros. Vale dizer que por usuário se entende o provedor de recursos de um dos federados da grade, sendo que indivíduos dentro de uma corporação federada são tratados como sendo a própria corporação.

Máquinas

Cada usuário u_i é proprietário de um conjunto M_i de K_i máquinas. O conjunto global de máquinas da grade computacional é M, formado pela união dos conjuntos de máquinas dos usuários, e contém K máquinas. Toda máquina m_{ij} do usuário u_i apresenta potência de c_{ij} Watts. Com isso, a potência da quota de um usuário u_i é denominada $PotênciaQuota_i$ e dada pela equação 3.1. das potências das máquinas que pertencem a ele. A potência total do sistema ($PotênciaSistema$) é calculada pela equação 3.2:

$$PotênciaQuota_i = \sum_{j=1}^{k_i} c_{ij} \tag{3.1}$$

$$PotênciaSistema = \sum_{i=1}^{N} \sum_{j=1}^{k_i} c_{ij} \tag{3.2}$$

Dentro do escopo deste projeto considera-se o consumo de energia, pelos recursos do sistema, como um parâmetro binário, ou seja, considera-se que a potência das máquinas é fixa e ocorre apenas quando o recurso estiver em uso, desprezando-se a potência em períodos de inatividade.

Além do consumo, cada máquina m_{ij} apresenta um poder computacional de d_{ij} MFLOPS. Sendo assim, o poder computacional da quota ($Quota_i$) de um usuário u_i é dado pela equação 3.3, enquanto o poder de processamento do sistema é obtido com a equação 3.4:

$$Quota_i = \sum_{j=1}^{k_i} d_{ij} \tag{3.3}$$

$$PoderSistema = \sum_{i=1}^{N} \sum_{j=1}^{k_i} d_{ij} \tag{3.4}$$

A eficiência energética de uma máquina m_{ij} é dada pela divisão de seu poder computacional pela sua potência, enquanto a eficiência média da quota de um usuário

é calculada dividindo o poder computacional de sua cota pela potência dela, conforme equação 3.5. Já a eficiência média do sistema com N usuários é dada pela equação 3.6:

$$EficiênciaQuota_i = \frac{\sum_{j=1}^{k_i} \frac{d_{ij}}{c_{ij}}}{k_i} \tag{3.5}$$

$$EficiênciaSistema = \frac{\sum_{i=1}^{N} \sum_{j=1}^{k_i} \frac{d_{ij}}{c_{ij}}}{N} \tag{3.6}$$

Limites de Potência e Potência Instantânea

Para todo usuário u_i se associa um limite de potência Lim_i, que é o teto de potência, em Watts, dado pelo usuário como porcentagem da potência total do conjunto de máquinas de u_i. Além do limite de potência é considerado também a potência instantânea PI_i de um usuário u_i em um dado instante de tempo. No instante considerado a potência instantânea é o somatório das potências das máquinas que estiverem executando tarefas de u_i. Por exemplo, se u_i está sendo atendido por três máquinas de 100 Watts cada, $PI_i = 300$ Watts.

Tarefas

Considera-se que as máquinas da grade executam uma tarefa por vez, e que, só existe fila de tarefas em espera no escalonador. Os usuários submetem conjuntos de tarefas individuais, ou seja, a fila do escalonador contém tarefas únicas e independentes entre si, em vez de grupos de tarefas cuja execução deve ser simultânea.

Cada usuário u_i submete um conjunto de tarefas T_i, com B_i tarefas. Toda tarefa t_{ik} pertence a u_i, tem um tamanho de s_{ik} MFLOP. Ainda para u_i, tarefas em execução pertencem ao conjunto P_i de R_i tarefas. O poder computacional total que atende a tarefas de um usuário u_i é $Alocado_i$, calculado pelo somatório dos poderes das máquinas que executam tarefas de u_i. As tarefas de u_i que estiverem em espera pertencem ao conjunto F_i de E_i tarefas. Com isso, um usuário u_i tem demanda não atendida se $F_i \neq \varnothing$.

Desperdício com preempção

Dada uma tarefa t_{ix} qualquer, que esteja sendo executada em uma máquina m_y qualquer, o desperdício de processamento em caso de preempção ($DesperdícioPreemp_y$), medido em MFLOP, é dado por meio da Equação 3.7 quando há *checkpointing* no processamento de tarefas, e pela Equação 3.8 quando não há:

$$DesperdícioPreemp_y = (TempoExec_{xy} \cdot d_y) \ mod \ BlocoCheckpoint \tag{3.7}$$

$$DesperdícioPreemp_y = TempoExec_{xy} \cdot d_y \tag{3.8}$$

Em ambas as equações $TempoExec_{xy}$ é a soma mais atualizada dos instantes em que a tarefa t_{ix} passou executando na máquina m_y, e d_y é o poder computacional de m_y em MFLOPS. Na equação 3.7 *BlocoCheckpoint* é o tamanho em MFLOP dos blocos de trabalho que são salvos no processo de *checkpointing*.

Quando não há *checkpointing* de tarefas o desperdício será todo o processamento feito, calculado multiplicando $TempoExec_{xy}$ por d_m, como é feito na Equação 3.8. Quando há *checkpointing* o desperdício é calculado pelo resto da divisão da quantidade de MFLOP processada, calculada conforme a 3.8, pelo tamanho do bloco de *checkpointing*. O resto da divisão corresponde ao montante de operações que não completou um bloco e será descartado.

3.2.3 Métricas de Decisão

Além dos parâmetros apresentados o EHOSEP utiliza algumas métricas para avaliar se continua ou não a alocação de novas tarefas. Várias dessas métricas derivam do HOSEP por tratarem mais especificamente de aspectos de justiça de propriedade, como poder remanescente esperado e diferencial de poder. A determinação algébrica dessas métricas é feita a seguir:

Coeficiente Sistema/Quota (CSQ)

Dado um usuário u_i, CSQ_i indica as relações entre o poder computacional de seus recursos e o poder do sistema, e também entre a potência da quota e a do sistema. Esse coeficiente é utilizado para tomadas de decisão com vistas a favorecer usuários que ofertarem máquinas eficientes do ponto . O cálculo do *CSQ* é dado pela Equação 3.9:

$$CSQ_i = \frac{PotênciaQuota_i}{PotênciaSistema} \cdot \frac{PoderSistema}{Quota_i} \tag{3.9}$$

Em que *PotênciaQuota* é o somatório das potências das máquinas do usuário, e *PotênciaSistema* é o somatório das potências das máquinas de todos os nós do sistema. Já *PoderSistema* é somatório de poder computacional de todas as máquinas do sistema, enquanto $Quota_i$ é o somatório de poder computacional das máquinas do usuário.

Diferencial de Poder

Dado um participante i da grade, atendido por um poder computacional correspondente a $Alocado_i$ MFLOPS e fornecedor de uma quota cujo desempenho total é de $Quota_i$ MFLOPS, calcula-se o diferencial de poder desse usuário como:

$$DP_i = \frac{Alocado_i - Quota_i}{Quota_i} \tag{3.10}$$

Como observado na Seção 2.4, o valor de DP_i será positivo se o usuário estiver usando mais poder computacional do que o que ofereceu. Será negativo caso contrário, sendo que o algoritmo tentará alocar novas tarefas desse usuário se ainda tiver demanda não atendida.

Poder Remanescente Esperado

O poder remanescente esperado é calculado para verificar o diferencial de poder resultante no caso de preempção. Dada uma máquina y, que está executando uma tarefa do usuário z, e que pode sofrer preempção para atender a uma tarefa do usuário w, calcula-se o poder remanescente esperado para u_z da seguinte forma:

$$PRE\_Preemp_{z,y} = \frac{Alocado_z - Quota_z - Poder_y}{Quota_z} \tag{3.11}$$

Definidos os parâmetros e métricas utilizados pelo algoritmo EHOSEP é possível fazer sua descrição mais precisa. O funcionamento do EHOSEP é apresentado no Algoritmo 3, sendo que para facilitar o seu entendimento detalhes da forma como tarefas em espera poderão ser atendidas são apresentados nos Algoritmos 4 e 5.

```
1  Faça AT = verdadeiro;
2  enquanto AT = verdadeiro faça
3  |  Faça i = 1;
4  |  Faça AT = falso;
5  |  Ordene a lista de usuários U em ordem crescente de DP;
6  |  enquanto (i ≤ N) ∧ (AT = falso) faça
7  |  |  se (PI_i < Lim_i) ∧ (E_i > 0) então
8  |  |  |  se ∃ máquina m_j livre | (c_j + PI_i) ≤ Lim_i então
9  |  |  |  |  Atendimento com máquina livre (Algoritmo 4);
10 |  |  |  |  Faça AT = verdadeiro;
11 |  |  |  senão
12 |  |  |  |  Tente atender com preempção (Algoritmo 5);
13 |  |  |  |  se atendeu com preempção então
14 |  |  |  |  |  Faça AT = verdadeiro;
15 |  |  |  |  fim
16 |  |  |  fim
17 |  |  fim
18 |  |  Faça i = i + 1
19 |  fim
20 fim
```

Algoritmo 3: EHOSEP

3.2.4 Detalhamento do algoritmo EHOSEP geral

O algoritmo geral do EHOSEP é controlado pela avaliação da variável AT, que indica se houve ou não alocação de uma nova tarefa para alguma máquina na iteração anterior. Deve ser observado que seu valor é inicialmente verdadeiro (**Linha 1**) para que seja possível tentar a primeira alocação. A seguir se descreve o funcionamento do algoritmo linha a linha.

Linha 2: enquanto ($AT = verdadeiro$) faça;

Propósito: repetir o procedimento de alocação de uma tarefa em uma máquina enquanto tiver ocorrido algum atendimento na iteração anterior, ou seja, a variável AT tiver valor *verdadeiro*.

Linha 4: Faça ($AT = falso$);

Propósito: alterar o valor de AT para o início do laço da linha 6. Como será visto, esse laço é repetido enquanto não houver atendimento de uma tarefa ($AT = falso$) e ainda existir usuário que não se tentou atender.

Linha 5: Ordene a lista de usuários U em ordem crescente de DP.

Propósito: ordenar a lista de usuários de modo que se tente atender primeiro o usuário com maior defasagem de recursos. Em caso de empate de DP, tem preferência o usuário u_a que oferecer maior desempenho (maior $\sum_{j=1}^{K_a} d_{aj}$). Se ainda houver empate, o usuário u_b cuja quota consumir menos energia (menor $\sum_{j=1}^{K_b} c_{bj}$), terá preferência. O primeiro critério de desempate favorece usuários que ofereçam mais poder computacional total à grade. Já o segundo critério de desempate dá preferência aos usuários que oferecerem recursos mais eficientes do ponto de vista energético.

Linha 6: enquanto $((i \leq N) \wedge (AT = falso))$ faça;

Propósito: controlar as tentativas de atendimento de uma nova tarefa. A primeira condição verifica se o algoritmo já tentou atender todos os usuários. A segunda condição verifica se já houve um atendimento ($AT = verdadeiro$), garantindo que o algoritmo saia do laço interno toda vez que conseguir alocar uma tarefa.

Linha 7: se $((PI_i < Lim_i) \wedge (E_i > 0))$ então;

Propósito: verificar se é possível alocar uma nova tarefa na grade. A segunda condição verifica se o usuário u_i ainda tem tarefa não atendida, aguardando por alocação. Já a primeira condição verifica se esse usuário já atingiu o limite de potência por ele definido (Lim_i), deixando de alocar tarefas deste usuário quando ele já tiver atingido esse limite. Apesar de essa condição induzir a que se defina limites altos de consumo, será visto adiante que o atendimento de uma tarefa que demande preempção favorece aos usuários com menor limite. Se não houver ociosidade no sistema, usuários com menor limite de potência perderão recursos apenas caso esse limite permita que ele utilize mais poder

computacional do que ofereceu. Tal comportamento do algoritmo é realista pois o sistema dificilmente permanecerá em ociosidade por longos períodos de tempo.

Linha 8: se ($\exists$ máquina m_j livre $\mid (c_j + PI_i) \leq Lim_i$) então;

Propósito: verificar se há máquinas livres cujo consumo de energia não faça o consumo total de u_i ultrapassar o limite por ele especificado.

Linha 9: Atenda com máquina livre (Algoritmo 4);

Propósito: alocar máquina livre para o usuário corrente u_i, seguindo o procedimento de alocação dado pelo Algoritmo 4, que é detalhado na Subseção 3.2.5. Após isso o valor de AT é modificado para indicar que houve a alocação de uma tarefa do usuário u_i.

Linha 12: Tente atender com preempção (Algoritmo 5);

Propósito: tentar atender u_i corrente com procedimento de preempção dado pelo Algoritmo 5, que é detalhado na Subseção 3.2.6.

Linhas 13 e 14: se (Atendeu com preempção) então Faça $AT = verdadeiro$;

Propósito: verificar se houve atendimento e, em caso afirmativo, alterar o valor de AT para que o laço interno seja encerrado.

3.2.5 Detalhamento do atendimento com máquina livre

Nesta subseção é explicado o Algoritmo 4, que é o componente do EHOSEP responsável por alocar uma tarefa, de um usuário que ainda consuma menos energia que seu limite, para uma máquina ainda livre.

1 Escolha a menor tarefa t_j de u_i que ainda não tenha sido alocada;
2 Escolha a máquina livre m_g compatível com o limite de potência de u_i, e que consumir menos energia na execução;
3 Aloca m_g para t_j;
4 Atualiza dados dos usuários e das tarefas;

Algoritmo 4: Atendimento com máquina livre no EHOSEP

Linha 1: Escolha a menor tarefa em espera t_j de u_i.

Propósito: escolher uma tarefa submetida pelo usuário a ser atendido tal que a tarefa escolhida seja aquela que apresentar o menor custo computacional entre as tarefas em espera do usuário u_i, ou seja:

$$t_j \mid t_j \in F_i \ \wedge \ (\forall \ t_a \in F_i \mid ((s_j \leq s_a) \ \wedge \ (t_a \neq t_j)))$$

Linha 2: Escolha a máquina livre m_g compatível com o limite de potência de u_i, e que consumir menos energia na execução.

Propósito: determinar qual máquina livre m_g executará a tarefa escolhida na linha 1. Essa máquina deve atender à condição do limite de potência de u_i, o que é expresso por $c_g + PI_i \leq Lim_i$, verificando se o consumo futuro do cliente, após a obtenção da máquina, será menor ou igual ao seu limite de potência. A máquina escolhida deve ser aquela que consumir menos energia para completar a tarefa entre aquelas que atendem às condições. Tal consumo é calculado dividindo o tamanho s_j da tarefa escolhida pelo poder computacional d_g da máquina testada, resultando no tempo necessário para execução, que é multiplicado pelo consumo c_g para que seja obtido o consumo total. Em caso de empate no consumo de energia total entre duas ou mais maquinas escolhe-se a máquina que tenha o maior poder computacional.

Linha 3: Aloca m_g para t_j;

Propósito: enviar a tarefa escolhida para execução à máquina m_g definida.

Linha 4: Atualiza dados dos usuários e das tarefas;

Propósito: atualizar as informações sobre o estado do sistema.

3.2.6 Detalhamento do atendimento com preempção

Nesta subseção é explicado o funcionamento do componente que faz a alocação de máquinas por preempção, que é ativado quando existirem tarefas habilitadas para execução e não existirem mais máquinas livres no sistema. Esse componente é apresentado no Algoritmo 5.

```
1  se u_i apresenta falta de poder computacional então
2      se existe u_z com excesso de poder computacional então
3          Escolha u_z com excesso de poder computacional e com o maior CSQ_z · PI_z;
4          se existe tarefa ativa de u_z executando em máquina compatível com o limite
             de potência de u_i então
5              Escolha para preempção a tarefa ativa t_y de u_z que executar em máquina
                 compatível com o limite de potência de u_i, e cuja interrupção apresentar
                 o menor desperdício de processamento;
6              se a preempção não levar a falta de poder computacional de u_z, ou se o
                 limite de potência de u_i for menor que o de u_z então
7                  Realiza preempção da tarefa t_y do usuário u_z;
8                  Aloca m_y para a tarefa em espera de menor tamanho, do usuário u_i;
9                  Atualiza dados dos usuários e tarefas;
10             fim
11         fim
12     fim
13 fim
```

Algoritmo 5: Atendimento por preempção no EHOSEP

Linha 1: Se u_i *apresenta falta de poder computacional* então;

Propósito: verificar se o usuário a ser atendido ocupa menos poder computacional do que oferece ao sistema, evitando assim que um usuário já privilegiado, em relação à sua quota, receba recursos em uso por outros usuários.

Linha 2: Se existe u_z com excesso de poder computacional então;

Propósito: verificar se há usuários com excesso de poder computacional ($DP_z > 0$), evitando-se que ocorra a preempção de recursos de usuários que não ocupem totalmente suas quotas.

Linha 3: Escolha u_z com excesso de poder computacional e com o maior $CSQ_z \cdot PI_z$;

Propósito: definir qual usuário deverá perder recursos por preempção. Esse usuário deverá ser aquele com maior valor de energia consumida em relação ao que oferece de poder computacional. Esse parâmetro é obtido multiplicando-se o potência instantânea do usuário PI_z por seu coeficiente sistema/quota, calculado pela Equação 3.9. Assim, usuários que tenham contribuição pequena e/ou que estejam consumindo energia em grande volume são os principais candidatos a perderem recursos por preempção. Em caso de empate entre dois ou mais candidatos escolhe-se aquele que tenha maior excesso de poder computacional naquele momento.

Linha 4: Se existe tarefa ativa de u_z executando em máquina compatível com o limite de potência de u_i então;

Propósito: verificar se o usuário u_z tem tarefas executando em máquinas cujo consumo respeita o limite de potência do usuário u_i que será beneficiado. Tal condição é expressa por $(\exists\ t_y \in P_z\ executada\ por\ m_y \mid (c_y + PI_i \leq Lim_i))$.

Linha 5: Escolha para preempção a tarefa ativa t_y de u_z que executar em máquina compatível com o limite de por de u_i, e cuja interrupção apresentar o menor desperdício de processamento;

Propósito: definir a tarefa de u_z que sofrerá preempção, que será aquela que, atendida as restrições anteriores, tiver o menor desperdício de processamento (Equações 3.7 e 3.8). Em caso de empate escolhe-se a tarefa executando em máquina com menor poder computacional.

Linha 6: Se a preempção não levar a falta de poder computacional de u_z, ou se o limite de por de u_i for menor que o de u_z então;

Propósito: avaliar as condições que permitem a preempção de um recurso, o que é possível se o usuário u_z ainda tiver excesso de poder computacional após a preempção, ou se, caso isso não ocorra, seu limite de potência for superior ao limite do usuário a ser beneficiado u_i. A verificação da existência de excesso ou falta de poder computacional do usuário que perderia o recurso em preempção é feita com a Equação 3.11, que no caso de falta de poder computacional resultaria em $PRE\_Preemp_{z,y} < 0$.

Linha 7: Realiza preempção da tarefa t_y do usuário u_z;

Propósito: interromper a execução da tarefa t_y, liberando a máquina m_y.

Linha 8: Aloca m_y para a tarefa em espera, de menor tamanho, do usuário u_i;

Propósito: alocar tarefa t_h do usuário u_i para a máquina m_y liberada na linha 7. A tarefa escolhida é aquela que apresentar o menor tamanho entre as tarefas inativas $(t_h \in F_i)$ de u_i, buscando minimizar o tempo médio de espera das tarefas submetidas.

Linha 9: Atualiza dados dos usuários e das tarefas;

Propósito: atualizar as informações sobre o estado do sistema.

3.3 Validação do critério de parada do EHOSEP

A eficiência de um algoritmo que permite preempção de tarefas é medida, em parte, pelo fato de não resultar em um processo circular de preempções. Isso significa na prática que o algoritmo deve ter um critério de parada claro e coerente com o propósito do algoritmo. Assim, nesta seção é verificado se o EHOSEP encerra sua execução em situações de sobrecarga, em que a tomada de um recurso levaria ao usuário que o detinha

a tomar outro recurso. Nos casos em que uma preempção gera defasagem para o usuário que perde o recurso, o algoritmo não deve, na iteração seguinte, repetir o procedimento retirando recursos de outros usuários ou, principalmente, daquele que acabou de receber.

Para que o EHOSEP encerre sua execução é necessário que o laço *enquanto* da linha 2 do algoritmo 3 termine, ou seja, a variável AT deve ser falsa após o término do laço da linha 7, indicando que o escalonador não pôde atender nenhum usuário. Na Expressão 3.12 estão dispostas as condições para a alocação de recursos para um usuário.

$$\begin{aligned} \exists\ atendimento\ \Rightarrow\ [\exists\ u_i:\ (PI_i\ <\ Lim_i\ \wedge\ E_i > 0)\ \wedge \\ [(\exists\ máquina\ livre\ m\ \mid\ c_m + PI_i\ \leq\ Lim_i)\ \vee \\ ((DP_i < 0)\ \wedge\ (\exists\ u_z\ \mid\ DP_z > 0)\ \wedge \\ (\exists\ tarefa\ t_y\ de\ u_z\ \mid\ t_y\ executa\ em\ m_y\ \wedge\ c_y + PI_i\ \leq\ Lim_i)\ \wedge \\ (PRE_{Preemp_{z,y}} \geq 0\ \vee\ (PRE_{Preemp_{z,y}} < 0\ \wedge\ Lim_i < Lim_z)))]] \end{aligned} \tag{3.12}$$

Para simplificar a análise do encerramento do EHOSEP as condições de atendimento serão renomeadas da seguinte maneira:

LNA (Limite Não Atingido): $PI_i\ \leq\ Lim_i$

ED (Existe Demanda): $E_i > 0$

EMLC (Existe Máquina Livre Compatível): $\exists\ máquina\ livre\ m\ \mid\ c_m + PI_i\ \leq\ Lim_i$

EF (Existe Falta): $DP_i < 0$

EE (Existe Excesso): $\exists\ u_z\ \mid\ DP_z > 0$

EMOC (Existe Máquina Ocupada Compatível): $\exists\ tarefa\ t_y\ de\ u_z\ \mid\ t_y\ executa\ em\ m_y \wedge c_y + PI_i\ \leq\ Lim_i$

PRENN (Poder Remanescente Esperado Não Negativo): $PRE_{Preemp_{z,y}} \geq 0$

VPL (Vantagem Por Limite): $Lim_i < Lim_z$

Utilizando a nova nomenclatura na expressão 3.12, obtida anteriormente, tem-se o seguinte:

$$\exists\ atendimento\ \Rightarrow\ \text{LNA}\ \wedge\ \text{ED}\ \wedge\ (\text{EMLC}\ \vee\ (\text{EF}\ \wedge\ \text{EE}\ \wedge\ \text{EMOC}\ \wedge\ (\text{PRENN}\ \vee\ \text{VPL}))) \tag{3.13}$$

Nesta avaliação, considera-se $Lim_i \geq \sum_{j=1}^{K} c_j$ e $E_i \geq K$, ou seja, o usuário em atendimento definiu um limite de potência que permite a alocação da grade inteira para

ele e, além disso, submete um número de tarefas que é, no mínimo, igual ao número de máquinas no sistema. A não existência de demanda ou o limite de potência ser ultrapassado são situações que levam a condições óbvias de parada, não necessitando de maior análise. Com isso, podem ser desconsideradas as condições LNA e ED pois serão sempre verdadeiras, o que leva à expressão 3.14.

$$\exists\ atendimento \Rightarrow \text{EMLC} \vee (\text{EF} \wedge \text{EE} \wedge \text{EMOC} \wedge (\text{PRENN} \vee \text{VPL})) \quad (3.14)$$

Dividindo a expressão em duas partes tem-se o seguinte:

$A = EMLC$

$B = \text{EF} \wedge \text{EE} \wedge \text{EMOC} \wedge (\text{PRENN} \vee \text{VPL})$

Utilizando esta representação tem-se que há atendimento se uma delas for verdade, ou $\exists\ atendimento \Rightarrow \text{A} \vee \text{B}$. Como o objetivo é o critério de parada, para que não haja atendimento de um usuário A e B devem ser falsos. No cenário considerado, em que $E_i \geq K$ e $Lim_i \geq \sum_{j=1}^{K} c_j$, EMLC será verdadeiro enquanto houver máquina livre, qualquer que seja a potência dela, pois o limite de potência de u_i permite alocar todas as máquinas da grade para o usuário. Entretanto, a quantidade de máquinas livres no sistema é de no máximo K, quando o sistema não estiver executando tarefa alguma. Conforme a alocação de máquinas livres for ocorrendo todas ficarão ocupadas, fazendo EMLC falso, e portanto A falso.

Assim, deixará de ocorrer atendimento de tarefas se B também se tornar falso quando não houver mais máquinas livres. Na análise de B sabe-se que para que seja falso é necessário que, no mínimo, EMOC ou EE ou EF ou PRENN e VPL juntos sejam falsos. Com $Lim_i \geq \sum_{j=1}^{K} c_j$ sempre haverá máquina com potência compatível para alocação, ou seja, EMOC sempre será verdadeiro, o que torna desnecessária a sua análise. Para os demais parâmetros se observa que:

i) EF será verdadeiro se o usuário que receberá uma máquina ocupar menos recursos do que ofereceu;

ii) EE será verdadeiro se existir usuário que esteja ocupando mais recursos do que ofereceu;

iii) VPL será verdadeiro se o limite de potência do usuário que receberá recursos for menor do que o limite do usuário que cederá recursos;

iv) PRENN será verdadeiro se o usuário que ceder recursos permanecer com poder remanescente esperado positivo (ainda tenha mais recursos do que ofereceu).

Supondo que B seja verdadeiro e não existam condições para alocar uma tarefa sem infringir as premissas do EHOSEP, então EE e EF são verdade ao mesmo tempo que ou PRENN ou VPL são verdade. Para que EF seja verdade é preciso que exista pelo menos um usuário com demanda ainda não atendida e que ainda não ocupe poder computacional equivalente ao que ofereceu. Para que EE seja verdade é preciso que exista pelo menos um usuário ocupando mais recursos do que ofereceu. Se esses valores forem verdade, então a premissa de justiça de propriedade ainda não foi atingida, permitindo a continuidade do escalonamento. Como a justiça de escalonamento é atingida quando não é mais possível trocar uma máquina entre dois usuários, então ou i) ou ii) se tornam falsas e o algoritmo atinge a condição de parada.

No caso da composição envolvendo VPL e PRENN, o que se quer verificar é se é possível uma delas ser verdade quando as premissas do algoritmo estiverem atendidas. Primeiro, caso a premissa de eficiência energética não seja válida, então VPL será falso. Depois, caso a premissa de justiça de propriedade tenha sido atingida, então PRENN será falso (não é possível tirar recurso de um usuário sem colocá-lo em situação de prejuízo). Assim, se as premissas forem atendidas, então é impossível que VPL ou PRENN sejam verdade, o que impede que B seja verdadeiro.

Um resultado indireto da análise sobre o valor de VPL é a constatação de que o algoritmo não entra em um ciclo infinito de preempção-realocação caso o usuário que tenha uma máquina retirada venha a recebê-la de volta na iteração seguinte. A hipótese para que isso ocorra é de que o usuário u_z que sofre preempção da máquina y passe a ter poder computacional negativo, sendo então escolhido para receber uma máquina. Caso isso venha a ocorrer, a condição examinada na Linha 6 do Algoritmo 5 estabelece que o usuário u_z receberá uma máquina apenas se seu limite de potência for menor que o limite de potência do usuário que sofrerá a preempção. Ocorre que isso é impossível se considerarmos o usuário que tenha recebido a máquina y, pois se recebeu aquela máquina então tinha limite de potência menor do que o limite de u_z. Assim, o algoritmo não entra em ciclo preempção-realocação, terminando efetivamente quanto atingir o estado de justiça de propriedade.

3.4 Critério de Justiça e Energia

Uma vez feita a análise do algoritmo EHOSEP é possível mostrar os estados em que o algoritmo colocará os usuários. De acordo com as linhas 2 e 6 do Algoritmo 3, o EHOSEP encerrará sua execução quando nenhum usuário puder ser atendido ($AT = falso$), indicando que uma das condições de não atendimento, vistas na validação do critério de parada do escalonador, ocorreu para cada um dos participantes da grade.

Os testes de potência feitos na linha 8 do algoritmo geral (Algoritmo 3), na

linha 2 do procedimento da alocação de máquinas livres (Algoritmo 4), e na linha 5 do procedimento de preempção (Algoritmo 5), garantem que o limite de potência dos usuários sempre será respeitado. Na alocação de máquinas sempre é verificado se a potência das máquinas consideradas para alocação, quando adicionada à potência instantânea do usuário beneficiado, levará a uma potência instantânea superior ao limite. Neste caso a alocação não ocorre.

A ordenação dos usuários feita na linha 5 do algoritmo geral (Algoritmo 3) e o teste $i \leq N$ na linha 6 do mesmo garantem que todos os usuários, desde que tenham demanda e não tenham atingido seus limites de potência, receberão recursos computacionais. Como a ordenação é feita sempre antes de um novo atendimento, o usuário beneficiado na última execução do laço iniciado na linha 2 terá um novo valor de Diferencial de Poder, e não ocupará a primeira posição da lista, sendo atendido na rodada seguinte apenas se os outros usuários não puderem receber máquinas, pois o teste da linha 6 garante que se o primeiro da lista não puder ser atendido, o EHOSEP tentará atender aos demais.

Sabendo que todos os usuários com demanda receberão recursos computacionais, considerando as condições de não atendimento dos usuários e comportamento do algoritmo, tem-se que o escalonador deixa os usuários em uma das seguintes condições:

i) o usuário é atendido por um poder computacional maior ou igual ao que ofereceu, e ele não tem demanda não atendida, ou seja, $DP_i \geq 0 \ \wedge \ E_i = 0$.

ii) o usuário é atendido por um poder computacional maior ou igual ao que ofereceu, mas ele tem demanda não atendida, ou seja, $DP_i \geq 0 \ \wedge \ E_i > 0$.

iii) o usuário é atendido por um poder computacional menor do que o oferecido por ele, entretanto não há tarefas dele em espera, ou seja, $DP_i < 0 \ \wedge \ E_i = 0$.

iv) o usuário é atendido por um poder computacional menor do que o oferecido por ele, e há tarefas dele em espera, ou seja, $DP_i < 0 \ \wedge \ E_i > 0$.

Nas condições *i*) e *iii*) os usuários não recebem mais recursos por não apresentarem tarefas em espera, ou seja, o atendimento para tais usuários é inibido pela condição de não atendimento $\neg ED$. Já a condição *ii*) refere-se ao caso em que o usuário tem tarefas em espera mas não recebe mais recursos ou por ter atingido seu limite de potência, ou por não haver máquinas livres de potência compatível com o limite definido pelo usuário. Além disso o usuário não apresenta falta de recursos, impedindo que seja beneficiado por preempções. Dessa forma a condição de não atendimento para a situação *ii*) é $(\neg EMLC \vee \neg LNA) \wedge \neg EF$.

Para a situação *iv*) a condição de não atendimento é expressa por $\neg EMLC \ \wedge$ $(\neg$EE $\vee$ $\neg$EMOC $\vee$ ($\neg$PRENN $\wedge$ $\neg$VPL)), ou seja, o usuário não pode receber máquinas livres ($\neg EMLC$) da mesma maneira que ocorre na condição ii, assim como não pode ser

beneficiado por preempções por não haver usuários com excesso de poder computacional ($\neg EE$), ou não haver máquina ocupada por tarefas de usuários com excesso de poder, e que tenha potência compatível com o limite do usuário a ser beneficiado. Mesmo no caso de existirem tais máquinas, se ocorresse a preempção o usuário prejudicado apresentaria falta de recursos ($\neg PRENN$), e o usuário beneficiado tem um limite de potência maior do que o prejudicado ($\neg VPL$).

3.5 Comparação do EHOSEP com outros algoritmos

Entre os escalonadores que consideram tanto energia quanto critérios de justiça, o EHOSEP é o único que usa a contribuição dada pelo usuário em grades cooperativas como parâmetro de decisão. Isso ocorre por meio da métrica de decisão CSQ (Equação 3.9), que relaciona a quota de recursos do usuário com a grade, tanto por seu poder computacional quanto por sua potência. Ao parametrizar essa relação com a potência instantânea durante a execução do procedimento de preempção (Algoritmo 5, linha 3), a métrica CSQ permite ao EHOSEP favorecer usuários que contribuem melhor com a grade e aqueles que limitam mais suas potências.

O EHOSEP considera em seu critério de justiça o poder computacional de cada usuário, enquanto o *MOSP-Energy*, que é o único algoritmo estudado que trata grades cooperativas, considera o custo operacional de cada participante, limitando o uso que um participante pode fazer de máquinas de outros usuários. Já os outros algoritmos vistos apresentam critérios de justiça inadequados para grades cooperativas, pois estão focados em dividir processamento e/ou energia de forma igualitária entre tarefas, ou trabalham para atender os requisitos de máquinas virtuais em sistemas de computação em nuvem.

Alguns dos algoritmos vistos utilizam DVFS para reduzir a velocidade de processamento, diminuindo a potência. Entretanto, o uso dessa técnica torna o escalonamento mais complexo, pois o escalonador deve decidir não apenas em que máquinas cada tarefa deve executar, mas em que par frequência/tensão a execução deve ocorrer, com eventual aumento no tempo de entrega das tarefas, o que não ocorre com o EHOSEP.

Outro aspecto a considerar é o histórico de uso do sistema. Os algoritmos *Energy Fair Share* e *MOSP-Energy*, por exemplo, necessitam da manutenção de históricos de uso para definir a potência permitida para a execução das tarefas, realizando tal definição com base nos seus respectivos critérios de justiça.

Já o tratamento do consumo com infraestrutura aparece apenas no escalonador *Power Credit based Fair scheduler*, que insere critérios adicionais ao processo de decisão ajudando a reduzir o consumo. Isso porém torna o escalonamento mais complexo e dependente de uma estrutura capaz de fornecer os dados necessários sobre o estado em que ela se encontra. Obter tais dados em uma grade cooperativa, normalmente formada de

recursos geograficamente dispersos não é simples de se fazer, sobrecarregando o escalonador.

3.6 Considerações Finais

O critério de alocação do EHOSEP combina propriedade de recursos com a eficiência energética dos mesmos. O algoritmo foi projetado para favorecer usuários que contribuem melhor com a grade, seja oferecendo grande volume de processamento, seja oferecendo recursos energeticamente eficientes, ou ainda encaixando-se nestes dois casos. Com isso espera-se estimular a formação de grades cooperativas, e desencorajar a oferta de recursos computacionalmente ineficientes apenas para se ter a chance de utilizar máquinas de outros participantes.

O EHOSEP mantém ainda o princípio de respeito à propriedade sobre os recursos que cada usuário oferece, ou seja, procura fazer com que cada participante seja atendido por um poder computacional maior ou igual ao que foi oferecido por ele. Tal premissa, combinada com os aspectos de energia considerados faz do EHOSEP um algoritmo único, diferenciando-se substancialmente dos algoritmos encontrados na literatura.

4 Testes e Resultados

Neste capítulo se apresenta a avaliação quantitativa do algoritmo EHOSEP. Essa avaliação é feita pela simulação do algoritmo em diferentes cenários, sendo também definida uma métrica de satisfação do usuário, desenvolvida para a avaliação do escalonador. Para a avaliação foram definidas algumas hipóteses a respeito do EHOSEP, correspondentes aos objetivos definidos para o algoritmo e à implementação dele, sendo as cargas de trabalhos das simulações estabelecidas a partir dessas hipóteses. Neste capítulo é apresentada inicialmente a métrica de satisfação do usuário na Seção 4.1 e, na sequência, as hipóteses e modelos avaliados na Seção 4.2 e os resultados das simulações realizadas na Seção 4.3.

4.1 Métrica de satisfação dos usuários

Uma vez que o EHOSEP busca favorecer usuários que forneçam mais e melhores recursos para a grade ou que evitem consumir mais energia, a métrica de satisfação dos usuários deve levar em conta esses aspectos. Isso significa que sua formulação deve tratar aspectos relativos ao consumo de energia, poder computacional recebido e perda de execução por preempções para favorecer outros usuários. A Equação 4.1, apresentada a seguir, compõe os parâmetros necessários para calcular a satisfação de um usuário u_i qualquer:

$$S_i = \left(100 \cdot \sum_{t=1}^{N_i} \frac{TempoIdeal_t}{TempoReal_t}\right) \cdot \left(\frac{ConsMaxQuota_i}{ConsTotalQuota_i}\right) \cdot \left(\frac{ConsSis - ConsTotal_i}{Limite_i \cdot TempoTotal_i}\right) \quad (4.1)$$

Os termos desta equação são os seguintes:

$\left(100 \cdot \sum_{t=1}^{N_i} \frac{TempoIdeal_t}{TempoReal_t}\right)$: soma das razões entre tempo de execução ideal ($TempoIdeal_t$) e o tempo de entrega real ($TempoReal_t$) das tarefas submetidas pelo usuário u_i. $TempoIdeal_t$ é o tempo de execução na máquina fictícia de poder computacional equivalente à media de poder do sistema. $TempoReal_t$ é o intervalo de tempo entre a submissão da tarefa e o término de sua execução, de forma que seja considerado também o tempo de espera da tarefa na fila do escalonador, com possíveis preempções.

$\left(\frac{ConsMaxQuota_i}{ConsTotalQuota_i}\right)$: razão entre o consumo máximo da quota de u_i ($ConsMaxQuota_i$), em Joules, pelo consumo total medido da quota (ConsTotalQuota_i), também em Joules. O consumo máximo é dado pelo tempo total de simulação multiplicado pelo consumo da quota em Watts. Com esse fator, a satisfação do usuário aumentará

conforme menos energia seja consumida com o uso de sua quota, pois isso levaria a um custo financeiro menor com energia.

$\left(\frac{ConsSis - ConsTotal_i}{Limite_i \cdot TempoTotal_i}\right)$: razão entre o consumo total dos demais usuários, dado pelo consumo total do sistema menos o consumo do usuário U_i, e o consumo efetivo deste usuário ponderado por seu limite de potência. Os valores de consumo total são obtidos pelo somatório dos consumos com a execução de cada tarefa na grade.

O primeiro termo da equação de satisfação tem por finalidade avaliar a satisfação do usuário com relação ao tempo de execução das tarefas. Para uma dada tarefa, quanto mais próximo o tempo real do tempo ideal, maior será a satisfação, pois a tarefa esperou menos tempo na fila e/ou foi executada por máquinas mais rápidas.

Já o segundo termo é uma medida do uso das máquinas fornecidas pelo usuário u_i, sendo um indicativo do custo da energia nelas consumida. Aqui a satisfação é maior se as máquinas forem pouco utilizadas. Como o algoritmo busca alocar tarefas em máquinas para as quais calcula-se que o consumo para executar a tarefas será menor, a composição desse termo com o anterior favorecerá usuários que forneçam máquinas com melhor valor de MFLOP/Watt.

Por fim, o terceiro termo considera a relação entre o limite de potência do usuário, o consumo de energia que ele gerou com suas tarefas, e o consumo total do sistema. A composição dos três termos considerados na métrica busca definir um valor global que indique se usuários que forneçam máquinas mais eficientes ou que permitam o uso reduzido de energia na execução de suas tarefas tenham maior valor de satisfação.

4.2 Hipóteses de avaliação e modelos de simulação

A avaliação do EHOSEP se deu por meio de simulação de alguns modelos de grade, buscando validar os objetivos do escalonador. A escolha por simulação se deu pela dificuldade em estruturar grades com diferentes composições de usuários e equipamentos, tanto de rede quanto de processamento. Para os testes utilizou-se o simulador de sistemas paralelos e distribuídos iSPD (*iconic Simulator of Parallel and Distributed Systems*) (53). A escolha pelo iSPD se deu em virtude da experiência prévia com ele, da robustez do simulador, e dos recursos que ele oferece para implementação de escalonadores, seja por meio de interface gráfica, seja com o uso de programação direta em Java.

Foram simulados dois modelos diferentes de grade, preparados com vistas a verificar as hipóteses derivadas dos objetivos do EHOSEP e de sua implementação. A seguir se descrevem as hipóteses de avaliação, os modelos utilizados e as cargas de trabalho aplicadas nas simulações.

4.2.1 Hipóteses

A definição do procedimento de testes partiu da definição de hipóteses que verificariam a efetividade do EHOSEP. Em especial foram consideradas três hipóteses para avaliação:

I) A satisfação dos usuários é proporcional ao poder computacional oferecido por eles;

II) Entre usuários com limite de potência e quotas de poder computacional similares, o EHOSEP favorece o usuário cujos recursos são mais eficientes energeticamente;

III) Entre usuários com quotas de poder computacional similares, o EHOSEP favorece o usuário com limite de potência mais baixo;

IV) Em condições equivalentes de carga o EHOSEP gera escalonamentos energeticamente mais econômicos que o HOSEP, sem perda significativa de desempenho;

Com a hipótese I) se busca mostrar que o algoritmo atende à premissa de justiça de propriedade, ou seja, garante que cada usuário conseguirá usar tanto poder computacional quanto oferecer. Já as hipóteses II) e III) tratam da premissa de consumo de energia, verificando o favorecimento de usuários que busquem essa redução por eficiência energética ou menor potência instantânea. Por fim, a hipótese IV) avalia se existe de fato economia no consumo de energia e o que isso significa do ponto de vista de desempenho.

4.2.2 Modelos de Teste

Os modelos utilizados são formados de máquinas ligadas diretamente a um escalonador central, utilizando *links* diretos de rede *gigabit*. As máquinas dos modelos apresentam poder computacional e potência de computadores reais, obtidos do *website* Tom's Hardware (54) (55).

Modelo 1

O primeiro dos modelos, chamado Modelo 1, consiste de um conjunto de 64 máquinas, oferecidas por quatro usuários chamados U1, U2, U3 e U4. Na Tabela 2 estão dispostos os dados sobre o modelo, mostrando o poder computacional e a potência das quotas de cada usuário no sistema. Esse modelo será usado para verificar as hipóteses I), II) e IV).

No Modelo 1, o usuário U1 contribui para a grade com o maior volume de processamento entre os usuários, e seus recursos apresentam eficiência energética mediana. Já U2 apresenta a quota mais eficiente, com poder computacional ainda significativo. U3 e U4 contribuem para a grade com volumes de processamento pouco expressivos, e semelhantes um ao outro, mas U4 apresenta máquinas muito mais eficientes que U3.

Tabela 2 – Características do Modelo 1.

Usuário	Número de Máquinas	Poder Computacional da máquina (GFLOPS)	Potência (W)	Poder total (GFLOPS)	Potência Total (W)	Eficiência Média (GFLOPS/W)
U1	16 8	133,47 108	220 191	2999,52	5048	0,5942
U2	12 8	84,53 95,49	99 113,5	1778,28	2096	0,8484
U3	6 6	32,42 17,38	134 118	298,8	1512	0,1976
U4	8	42,15	67	338	536	0,6306
Total	64	—	—	5414,6	9192	0,589

Fonte: do autor

Modelo 2

O segundo modelo é formado de 64 máquinas, também ligadas diretamente a um escalonador central por *links* diretos de rede *gigabit*. As máquinas são oferecidas pelos usuários U1, U2 e U3, conforme pode ser visto na Tabela 3. Neste modelo, os participantes oferecem volumes de processamento muito próximos, mas diferem muito em eficiência energética. U2 é o usuário de recursos mais eficientes, seguido de U1 e de U3, com este último oferecendo recursos de eficiência muito baixa. Essa diferença de eficiência energética permite a verificação das hipóteses II) e III).

Tabela 3 – Características do Modelo 2

Usuário	Número de Máquinas	Poder Computacional da máquina (GFLOPS)	Potência (W)	Poder total (GFLOPS)	Potência Total (W)	Eficiência Média (GFLOPS/W)
U1	10	133,47	220	1334,7	2200	0,6067
U2	14	95,49	113,5	1336,86	1589	0,8413
U3	40	32,42	134	1296,8	5360	0,2419
Total	64	—	—	3968,6	9149	0,4338

Fonte: do autor

4.2.3 Carga de trabalho simulada

Nas simulações feitas com os modelos definidos anteriormente é aplicada uma carga aleatória dentro de determinados padrões. As cargas aplicadas são compostas por conjuntos de tarefas computacionalmente intensivas, submetidas pelos diferentes usuários, em momentos pré-estabelecidos. Considera-se aqui que tarefas mais curtas teriam impacto negligível no consumo de energia, por isso a escolha por tarefas de maior duração.

Na Tabela 4 é apresentado o valor médio de poder computacional de cada modelo, enquanto na Tabela 5 estão apresentados os tipos de tarefas que compõem a carga de trabalho, definidos de acordo com o tempo de execução em máquina fictícia de poder computacional equivalente às médias da Tabela 4. Dessa forma, ficam definidos os tamanhos para cada tipo de tarefa nas Tabelas 6 e 7, nos modelos 1 e 2 respectivamente.

Tabela 4 – Máquinas médias dos modelos

Modelo	Poder computacional médio das máquinas (GFLOPS)
1	61,01
2	84,6

Fonte: do autor

Tabela 5 – Tempos de execução dos tipos de tarefa

Tipo de tarefa	Tempo mínimo na máquina média do modelo	Tempo máximo na máquina média do modelo
Tipo A	10 minutos	30 minutos
Tipo B	30 minutos	50 minutos
Tipo C	50 minutos	70 minutos

Fonte: do autor

Tabela 6 – Tamanhos de tarefa do Modelo 1

Tipo de Tarefa	Tamanho Mínimo (GFlop)	Tamanho Máximo (GFlop)
Tipo A	36606	109818
Tipo B	109818	183030
Tipo C	183030	256242

Fonte: do autor

Tabela 7 – Tamanhos de tarefa do Modelo 2

Tipo de tarefa	Tamanho mínimo (GFlop)	Tamanho máximo (GFlop)
Tipo A	50760	152280
Tipo B	152280	253800
Tipo C	253800	355320

Fonte: do autor

Os tamanhos das tarefas foram definidos de forma que as do Tipo A sejam pequenas, as do tipo B médias e as do tipo C sejam grandes. O uso dos intervalos de tempo considerados para calcular os tamanhos das tarefas tem por objetivo torná-los dependentes do modelo, e a variação deles dentro dos intervalos é feita aleatoriamente.

A quantidade de tarefas que cada usuário submete nas simulações de cada modelo pode ser vista nas Tabelas 8 e 9. O número de tarefas submetido por usuário foi definido de

forma que seja proporcional à quantidade de recursos de cada participante, e a distribuição dos tipos de tarefa na submissão de cada usuário foi inspirada na distribuição Normal de probabilidade. Além disso, com essas quantidades de tarefa pretende-se simular a utilização plena dos sistemas, promovendo competição entre os usuários pelos recursos, o que é necessário para avaliar o atendimento de seus objetivos pelo EHOSEP.

Tabela 8 – Tarefas submetidas pelos usuários do Modelo 1

Usuário	Número de Tarefas A	Número de Tarefas B	Número de Tarefas C
U1	25	50	25
U2	20	40	20
U3	15	30	15
U4	15	30	15

Fonte: do autor

Tabela 9 – Tarefas submetidas pelos usuários do Modelo 2

Usuário	Número de Tarefas A	Número de Tarefas B	Número de Tarefas C
U1	10	20	10
U2	14	28	14
U3	40	80	40

Fonte: do autor

Em ambiente real normalmente haveriam mais tarefas pequenas e poucas tarefas grandes, que seriam responsáveis por quase a totalidade do tempo de atividade do sistema (56). Entretanto, aplicou-se neste trabalho uma carga de trabalho que levaria na prática a uma maior pressão sobre o escalonador com relação ao consumo de energia, criando uma espécie de pior caso para avaliar o EHOSEP.

4.2.4 Sequências de teste

Como a verificação da hipótese IV) implica na avaliação de desempenho do algoritmo, a mesma será apresentada em separado, na Seção 4.4. Para as demais hipóteses aplicou-se algumas sequências de teste, descritas na próxima seção, e em cada uma delas a submissão das tarefas de um dos usuários é atrasada em 10 minutos, enquanto os demais usuários têm suas tarefas submetidas no início da simulação, quando todas as máquinas do modelo estão ociosas. Cada sequência de teste foi feita com e sem *checkpointing* de tarefas, configurado para conservar o progresso da execução das tarefas em blocos de 5 minutos de execução. Além disso, o limite de potência do usuário atrasado varia ao longo dos casos de teste em cada sequência.

As sequências possuem quatro casos de teste cada uma, e em cada um deles o limite de potência do usuário atrasado muda, enquanto os outros usuários apresentam limites absolutos equivalentes à potência total do sistema. No primeiro caso o limite do usuário atrasado é de 80%, no segundo é de 100%, no terceiro é de 150%, e no último é a porcentagem que corresponde ao valor absoluto equivalente à potência total do sistema. Estas porcentagens estão dispostas na Tabela 10.

Combinando o atraso na submissão de tarefas com a variação no limite de potência do usuário que as submete espera-se: verificar como o EHOSEP tira recursos dos demais usuários para atender ao usuário atrasado; e como esse atendimento se modifica conforme muda o limite de potência do usuário atrasado.

Cada caso de teste das sequências foi simulado 60 vezes. Tal número de simulações foi considerado adequado por meio de testes de convergência estatística, que foram feitos com base no desvio padrão das médias, calculadas após adição dos resultados conforme as simulações eram feitas. A lista a seguir enumera as sequências de testes realizadas.

S1M1 CC/SC: simulação do Modelo 1, com *checkpointing* (CC) e sem ele (SC), atrasando a submissão das tarefas do usuário U1.

S2M1 CC/SC: simulação do Modelo 1, com *checkpointing* (CC) e sem ele (SC), atrasando a submissão das tarefas do usuário U2.

S3M1 CC/SC: simulação do Modelo 1, com *checkpointing* (CC) e sem ele (SC), atrasando a submissão das tarefas do usuário U3.

S4M1 CC/SC: simulação do Modelo 1, com *checkpointing* (CC) e sem ele (SC), atrasando a submissão das tarefas do usuário U4.

S1M2 CC/SC: simulação do Modelo 2, com *checkpointing* (CC) e sem ele (SC), atrasando a submissão das tarefas do usuário U1.

S2M2 CC/SC: simulação do Modelo 2, com *checkpointing* (CC) e sem ele (SC), atrasando a submissão das tarefas do usuário U2.

S3M2 CC/SC: simulação do Modelo 2, com *checkpointing* (CC) e sem ele (SC), atrasando a submissão das tarefas do usuário U3.

4.3 Resultados

São apresentados a seguir os resultados obtidos nos testes. Para simplificar a avaliação serão usados gráficos de satisfação dos usuários (equação 4.1). Em todos os resultados foi calculado o intervalo de confiança considerando certeza de 95%. Os números

Tabela 10 – Limites para ocupar todo o sistema

Usuário	Modelo 1	Modelo 2
1	182,09 %	415,87%
2	438,55 %	575,78%
3	607,94 %	170,7%
4	1714,93 %	—

Fonte: do autor

apresentados nos gráficos são as médias do que foi obtido nos 60 testes feitos no caso particular a que o gráfico se referir.

4.3.1 Resultado S1M1 CC/SC

Com *Checkpointing*

Na sequência S1M1 com *checkpointing* (CC), a satisfação dos usuários variou conforme é apresentado no gráfico da Figura 2, relacionando o limite de potência do usuário atrasado U1 com a satisfação de cada participante da grade simulada.

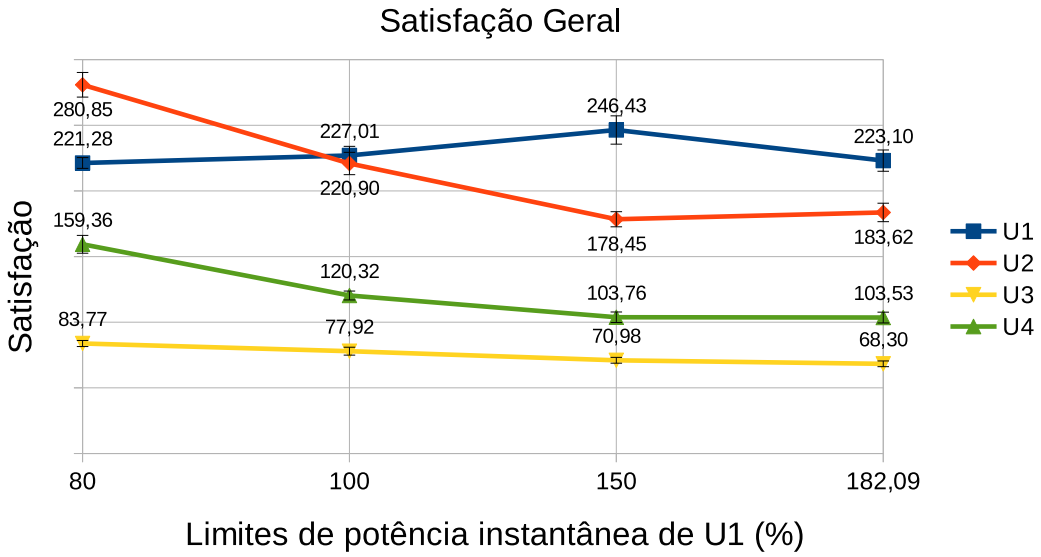

Figura 2 – Satisfação dos usuários (S1M1CC) com potência instantânea de U1 limitada e dos demais ilimitada

Neste gráfico é possível notar que quando U1 limita sua potência instantânea em 80% da de sua quota, ele fica menos satisfeito que U2, que oferece menos recursos computacionais que U1. Isso ocorre porque com o limite de 80% U1 é atendido por menos máquinas, fazendo com que seu conjunto de tarefas demore mais para executar. Por outro lado, U2 fica muito satisfeito, pois utiliza máquinas livres que U1 não ocupou.

Conforme U1 vai aumentando seu limite de potência, sua satisfação melhora até que este ultrapasse 150%. Isso ocorre porque, a partir deste ponto, U1 passa a ter um limite de potência idêntico, em valor absoluto, aos demais. Com isso, U1 perde a vantagem que tinha por limitar mais sua potência instantânea do que os outros usuários.

A partir do momento em que U1 limita sua potência instantânea em 100%, a satisfação de cada usuário fica proporcional ao poder computacional que ofereceu, e também depende da eficiência dos recursos. Ao longo do teste, U3 e U4 apresentam satisfações diferentes, mesmo oferecendo volumes computacionais semelhantes entre si. O motivo para isso é a maior eficiência energética das máquinas de U4. Considerando o que foi observado no gráfico de satisfação, neste teste ficam confirmadas as hipóteses I) e II).

Sem *Checkpointing*

Na Figura 3 encontra-se o gráfico de satisfação para a sequência de teste S1M1 sem *checkpointing* de tarefas. Os resultados são muito semelhantes aos obtidos com *checkpointing* e, portanto, sustentam as mesmas hipóteses que foram confirmadas no teste anterior. Além disso, fica claro que o uso de *checkpointing* não teve grande impacto sobre a experiência dos usuários com o sistema e, portanto, a partir de agora, serão apresentados apenas os resultados dos testes sem *checkpointing*. Isso evita um excesso de informações neste capítulo e a escolha pelos testes sem *checkpointing* se dá pois a ausência dele torna preempções mais prejudiciais, exigindo mais do EHOSEP para tornar os usuários satisfeitos. Os resultados dos testes com *checkpointing* podem ser vistos no Apêndice A.

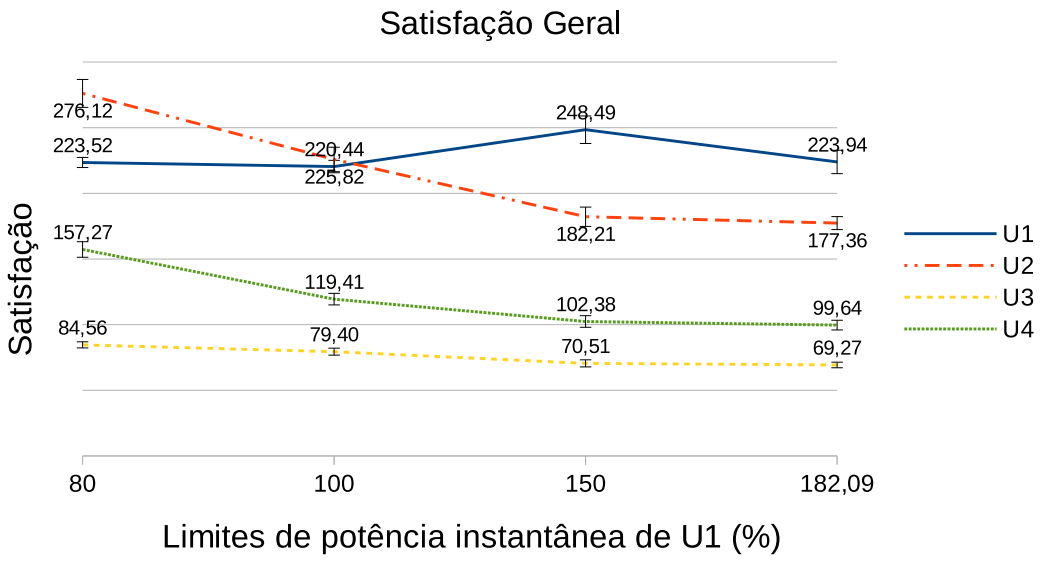

Figura 3 – Satisfação dos usuários (S1M1SC) com potência instantânea de U1 limitada e dos demais ilimitada

Uso do sistema

Na figura 4 é apresentado um gráfico de uso do poder computacional da grade pelos usuários. Este gráfico foi obtido de uma simulação única do caso de teste S1M1SC, pois ainda não há meios de obter um gráfico de uso médio do sistema com o iSPD.

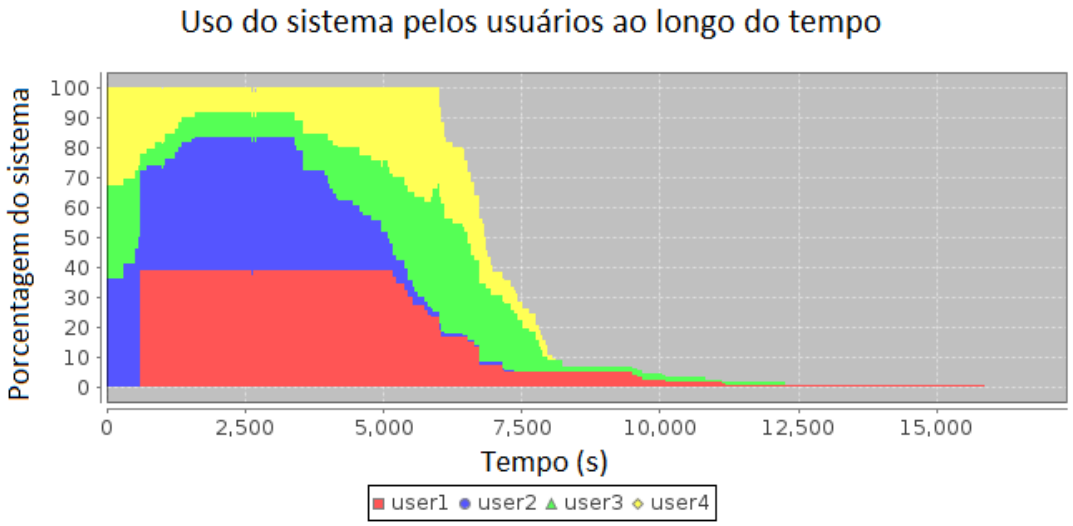

Figura 4 – Distribuição do poder computacional entre os usuários

Por meio do gráfico observa-se que a carga de trabalho submetida é suficiente para ocupar todo o sistema por um período de tempo significativo, e que as porcentagens de uso do sistema pelos usuários correspondem ao poder computacional que ofereceram. Observa-se também que assim que U1 submete suas tarefas o EHOSEP aloca para ele o poder computacional correspondente a sua quota, diminuindo os recursos em uso pelos outros usuários. Também é observável o longo tempo de execução de algumas tarefas de U1, o que ocorre devido à alocação de tarefas grandes deste usuário para máquinas de baixo poder computacional em procedimentos de preempção. O cenário observado no gráfico de uso repetiu-se em todos os testes, e portanto não serão apresentados outros gráficos de uso para os testes seguintes.

4.3.2 Resultado S2M1 SC

O gráfico apresentado na Figura 5 mostra como variou a satisfação dos usuários, conforme o limite de U2 mudou ao longo dos testes na sequência S2M1, sem *checkpointing* de tarefas. No gráfico observa-se que enquanto o limite de U2 permanece menor que o dos outros usuários, sua satisfação aumenta chegando mesmo a ultrapassar a satisfação de U1, cuja quota tem maior poder computacional.

Quando o limite de U2 se iguala, em valor absoluto, ao limite dos outros participantes, a satisfação de U2 passa a ser menor que de U1, e proporcional ao poder computacional que ofereceu. Além disso, durante todo o teste, as satisfações de U1, U3 e

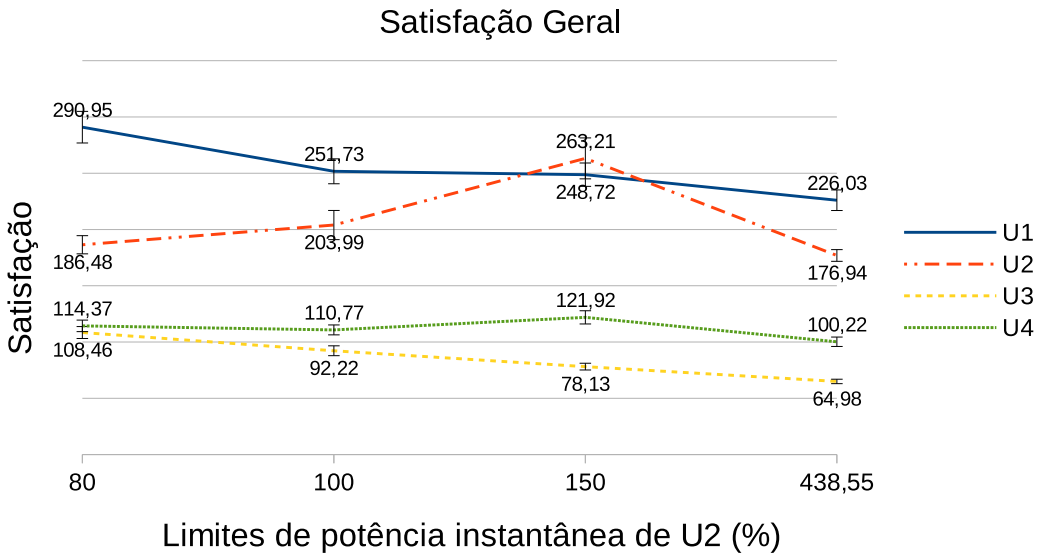

Figura 5 – Satisfação dos usuários (S2M1SC) com potência instantânea de U2 limitada e dos demais ilimitada

U4 se mantiveram proporcionais às quotas, com U4 mais satisfeito que U3 devido à maior eficiência de seus recursos. Ficam confirmadas mais uma vez as hipóteses I) e II).

4.3.3 Resultado S3M1 SC

Na sequência de testes S3M1SC, a satisfação dos usuários variou conforme apresentado na Figura 6. Por meio daquele gráfico nota-se que U3, quando limita sua potência instantânea em 80%, fica mais satisfeito que U1. Isso ocorre porque U3, ao tentar economizar energia, perde poucas tarefas por preempção, retendo eventualmente máquinas de outros usuários. Como essas máquinas são muito mais poderosas do que as de U3, sua satisfação aumenta.

Conforme U3 vai aumentando seu limite de potência, sua satisfação cai até o nível proporcional à sua quota, com a satisfação de U2 crescendo um pouco, enquanto as de U1 e U4 tem pequena queda. Isso mostra que ao entrar depois no sistema, mas com limite de potência maior, U3 deixa de retirar máquinas de U2, que tem as máquinas energeticamente mais eficientes. Esse resultado também valida a hipótese II), embora U2 tenha poder computacional diferente de U1 e de U3. Por fim, a hipótese I) é novamente verificada pela proporcionalidade entre quotas e satisfação do usuário.

4.3.4 Resultado S4M1 SC

Na Figura 7, pode ser visto o gráfico de satisfação dos usuários na sequência de testes S4M1. Pelo gráfico nota-se que a satisfação de todos os usuários começa com valores

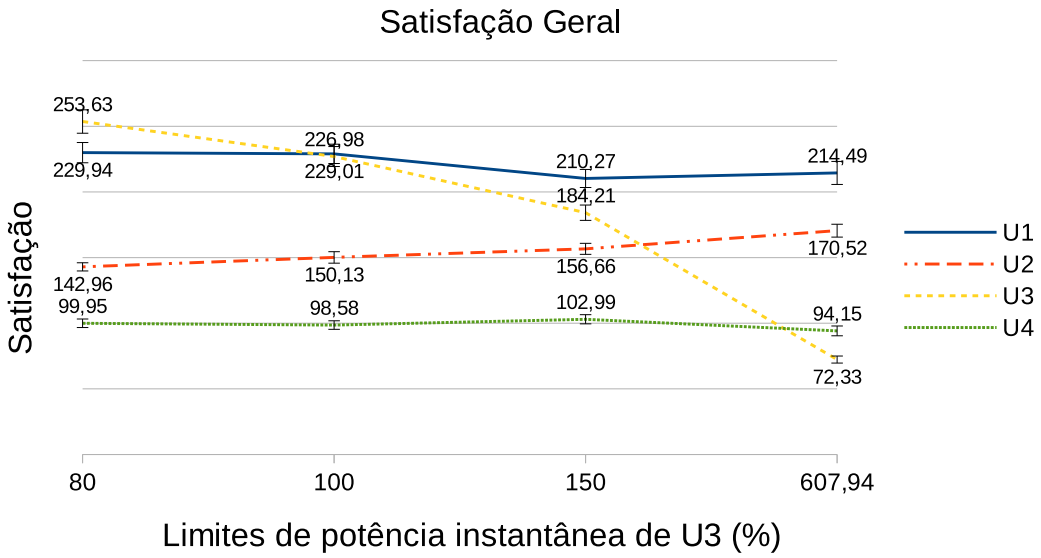

Figura 6 – Satisfação dos usuários (S3M1SC) com potência instantânea de U3 limitada e dos demais ilimitada

mais elevados, em relação ao que foi observado nos outros resultados. Isso ocorre porque a quota de U4 é pequena e com limite de potência baixo, fazendo com que os limites dele sejam naturalmente baixos também. Com isso, U4 consegue fazer a preempção de poucas máquinas, deixando mais máquinas para uso dos demais usuários, que obviamente ficam mais satisfeitos.

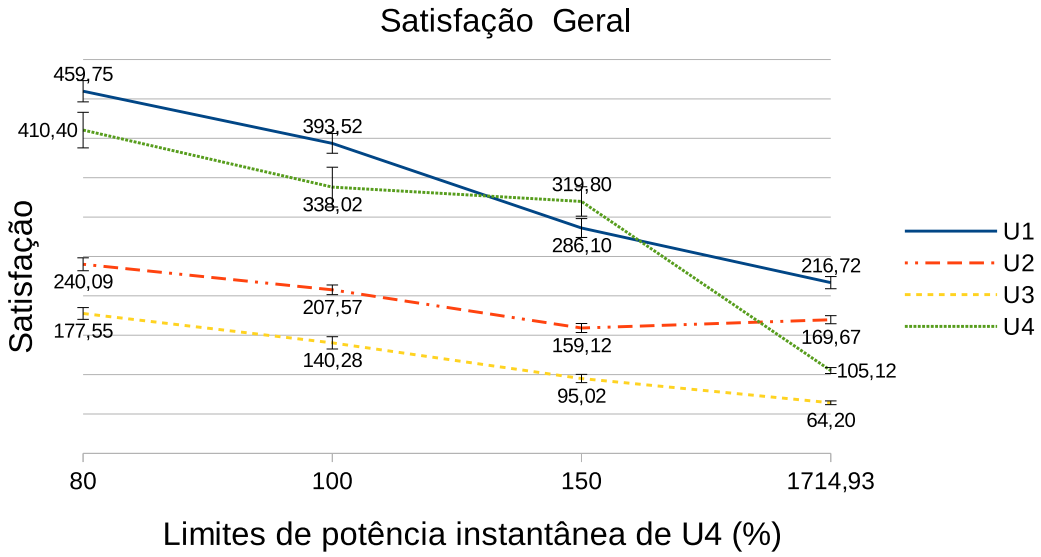

Figura 7 – Satisfação dos usuários (S4M1SC) com potência instantânea de U4 limitada e dos demais ilimitada

Mesmo assim, para limites de potência menores (até 150%), U4 ainda apresenta satisfação maior que a de U2. O motivo para isso está no fato de U4 perder poucas

máquinas após o sistema atingir o atendimento dele com a sua quota. Ou seja, o EHOSEP permite que U4 retenha máquinas, entre as quais estão, eventualmente, máquias de outros usuários, com elevado poder computacional.

Mais uma vez, observa-se a proporcionalidade das satisfações, com relação às quotas, com o limite mais alto de U4, confirmando a hipótese I). Os resultados também mostram que o EHOSEP favorece muito os usuários que apresentam recursos eficientes, confirmando a hipótese II).

4.3.5 Resultado S1M2 SC

Passando agora aos resultados dos testes com o Modelo 2, em que os usuários oferecem quotas de poder computacional idêntico, mas que variam muito em eficiência energética. Esse modelo tem por objetivo verificar as hipóteses II) e III). Para essa primeira sequência o usuário U1 começa a submeter tarefas depois dos demais usuários.

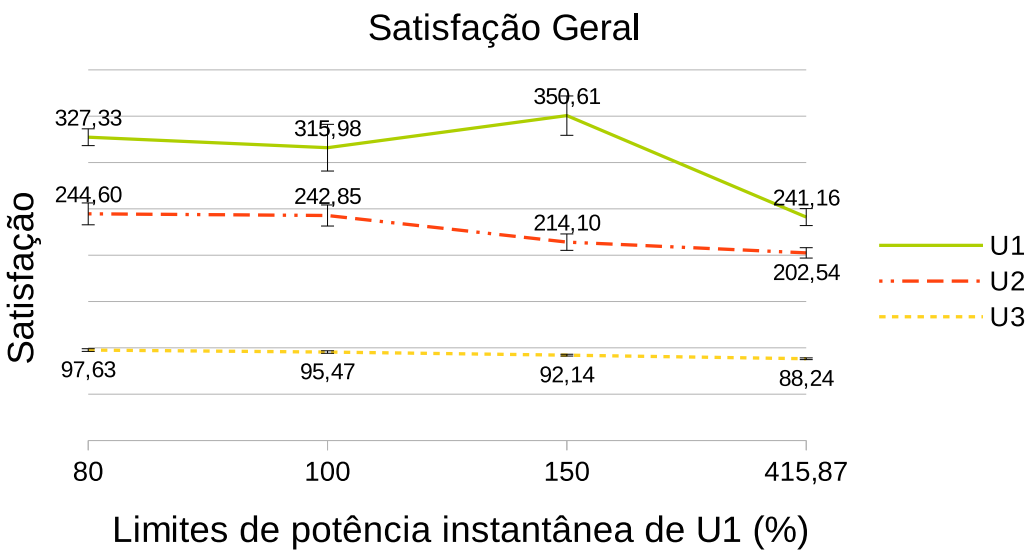

Figura 8 – Satisfação dos usuários (S1M2SC) com potência instantânea de U1 limitada e dos demais ilimitada

Tem-se na Figura 8 o gráfico de satisfação dos usuários, em que se verifica que U2 está sempre mais satisfeito que U3, o que ocorre devido à maior eficiência dos recursos de U2, ou seja, U3 sofre mais preempções do que U2. Apesar de U1 ter máquinas menos eficientes que U2, ele fica mais satisfeito que este em virtude das limitações de potência instantânea e do atraso de suas tarefas, que leva a uma baixa quantidade de tarefas interrompidas de U1. Com isso as hipóteses II) e III) estão confirmadas.

4.3.6 Resultado S2M2 SC

Para a sequência S2M2SC de testes, a satisfação dos usuários é apresentada no gráfico da Figura 9. Nele se nota que, apesar das máquinas de U1 serem menos eficientes que as de U2, este fica menos satisfeito que aquele. Isso ocorre porque as máquinas de U1 apresentam maior poder computacional, elevando sua satisfação e mostrando que o favorecimento do EHOSEP quanto à limitação da potência instantânea não sobrepuja diferenças significativas no poder computacional das máquinas.

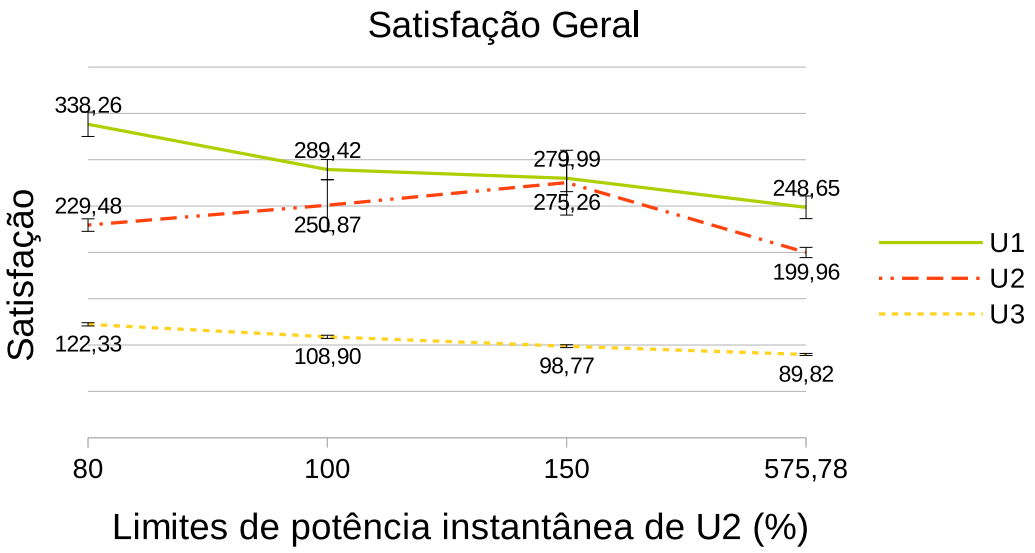

Figura 9 – Satisfação dos usuários (S2M2SC) com potência instantânea de U2 limitada e dos demais ilimitada

Entretanto, a hipótese II) não fica anulada pois U1 e U2 ficam muito mais satisfeitos que U3, cuja quota apresenta eficiência energética muito baixa. Por outro lado, a hipótese III) se verifica apenas quando U2 limita sua potência instantânea em 150%. Quando U2 limita sua potência instantânea em 80% e 100% ele é favorecido pelo escalonador, mas tal limitação no contexto do modelo impede que U2 receba poder computacional suficiente para ficar mais satisfeito que U1.

4.3.7 Resultado S3M2 SC

No gráfico da Figura 10 estão dispostas as satisfações dos usuários na sequência de testes S3M2SC. Nota-se pelo gráfico que conforme o usuário U3 atrasado aumenta seu limite de potência, a satisfação de U1 cai, indicando que este participante estava tirando o maior proveito das máquinas ociosas de U3. Além disso, U1 e U2 ainda ficam mais satisfeitos que U3, confirmando a hipótese II).

Observa-se ainda que a hipótese III) não se confirma. Apesar de U3 limitar sua potência instantânea muito mais que U2 e U1, o poder computacional muito baixo de seus

recursos leva à baixa baixa satisfação do usuário. Além disso a baixa eficiência energética das máquinas faz com que o EHOSEP desfavoreça U3 em preempções, prejudicando a satisfação desse usuário. O que se observa então é que o EHOSEP não estimula a participação de usuários com recursos de baixo poder computacional e eficiência energética pois, mesmo que limitem suas potência instantânea, esses usuários permanecem com baixa satisfação.

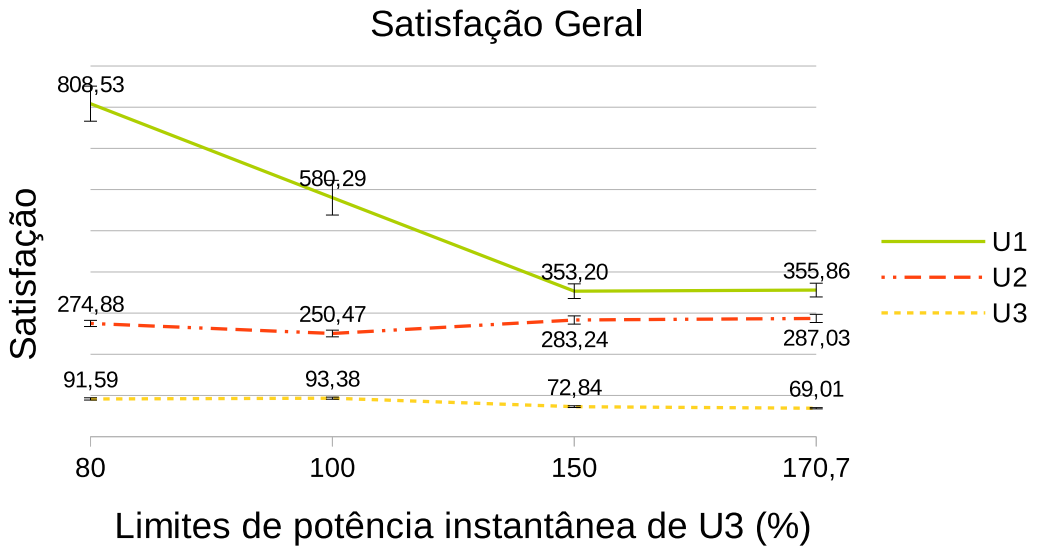

Figura 10 – Satisfação dos usuários (S3M2SC) com potência instantânea de U3 limitada e dos demais ilimitada

4.4 Avaliação de Desempenho

A métrica de satisfação dada pela equação 4.1 condensa os aspectos fundamentais tratados pelo EHOSEP, porém não indica, por exemplo, se houve de fato redução no consumo de energia. Nesta seção se faz a avaliação isolada do consumo de energia e do desempenho computacional da grade. Isso é feito pela comparação dos resultados da aplicação do EHOSEP com os do HOSEP, medindo-se o consumo de energia em joules por segundo (KJ/s) e o desempenho considerando-se o tempo de entrega médio das tarefas (*Turnaround time* médio). A carga de trabalho simulada nos testes é a mesma aplicada nos testes anteriormente apresentados.

O primeiro teste consistiu em simular o Modelo 1 apresentado anteriormente, sem *checkpointing* e sem atrasar a submissão de tarefas. Aqui, entretanto, todos os usuários tiveram limites de potência impostos, usando as mesmas faixas já utilizadas. Para o primeiro caso deste teste todos os usuários aplicam o limite de 80%, enquanto nos casos subsequentes os limites são 100%, 150% e a porcentagem correspondente à potência de

todo o sistema (Tabela 10). Os resultados deste teste são comparados com o resultado que foi obtido na simulação do mesmo modelo aplicando o algoritmo HOSEP.

Já para o segundo teste foi simulado o Modelo 1, também sem *checkpointing*, mas atrasando a submissão de tarefas de U2 e variando o seu limite da mesma forma com que foram feitos os testes anteriormente apresentados neste capítulo, ou seja, no primeiro caso U2 apresenta limite de 80%, no segundo 100%, no terceiro 150% e no último o limite correspondente à potência total do sistema, enquanto os outros usuários sempre aplicam o limite correspondente à potência total do sistema. Os resultados deste segundo teste foram comparados com os números obtidos na simulação do Modelo 1 aplicando o escalonador HOSEP.

A métrica de *Turnaround time* médio foi calculada por meio da média dos *Turnaround times* de cada tarefa, que por sua vez foi calculo subtraindo o instante de tempo em que a execução da tarefa termina, do instante de tempo em que a tarefa foi submetida ao sistema. Já o consumo de energia foi calculado pela divisão do consumo total de cada usuário, em Joules, pelo tempo de simulação, obtendo um valor em KJoules/segundo. Os resultados apresentados a seguir são as médias de 60 testes feitos para cada caso analisado, e nos gráficos foi colocado o intervalo de confiança para 95% de certeza.

4.4.1 Teste sem atraso de usuários

Neste teste foi simulado o Modelo 1 (Seção 4.2.2) sem que fosse atrasada a submissão de tarefas dos usuários. Os casos de teste foram quatro, em que todos os usuários aplicaram o mesmo limite de potência, sendo os limites do primeiro para o último caso os seguintes: 80%, 100%, 150% e Total do sistema (Tabela 10). Todos os casos de teste foram simulados 60 vezes e a média dos resultados, com os respectivos intervalos de confiança são apresentados nos gráficos a seguir.

No gráfico da Figura 11 estão dispostos os consumos dos usuários conforme se variou o limite de potência aplicado por eles nos testes em que utilizou-se o EHOSEP. Já na Figura 12 está disposto o gráfico com o consumo de energia dos usuários nos testes com o HOSEP. Por meio da Tabela 11 é possível verificar como o consumo dos usuários e do sistema (Total) nos testes com o EHOSEP variou com relação aos valores encontrados nos testes com o HOSEP, sendo que valores negativos indicam que o consumo foi menor com o EHOSEP.

O gráfico da Figura 13 apresenta a variação do *turnaround time* médio dos usuários conforme alterou-se o limite de potência dos usuários, na simulação do Modelo 1 com a aplicação do EHOSEP. Já para os testes com HOSEP, o *turnaround time* médio apresentou-se conforme o gráfico da Figura 14. A Tabela 12 estabelece a relação entre os valores de *turnaround time* médio obtidos nos teste com HOSEP e EHOSEP, sendo que valores

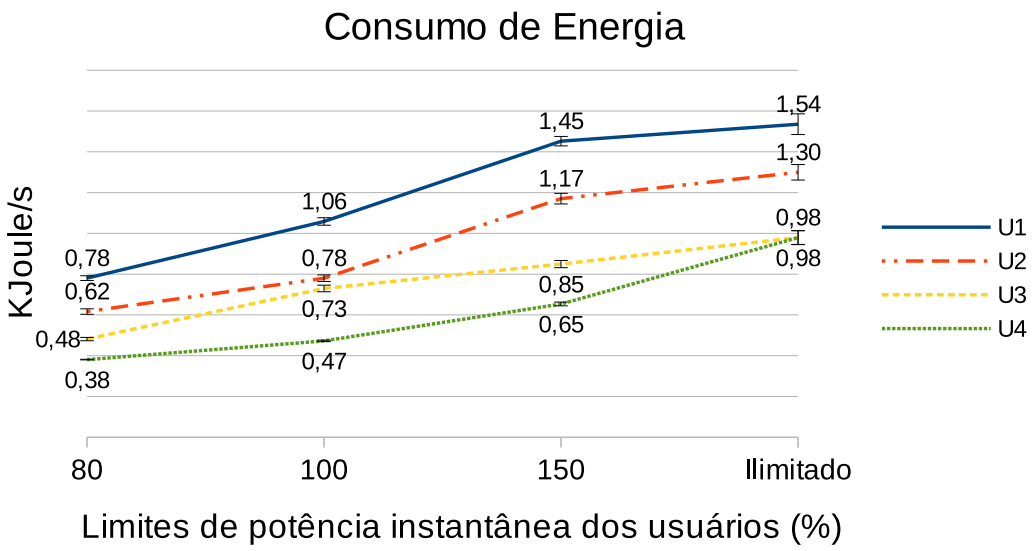

Figura 11 – Consumo dos usuários com todos aplicando limites, utilizando EHOSEP

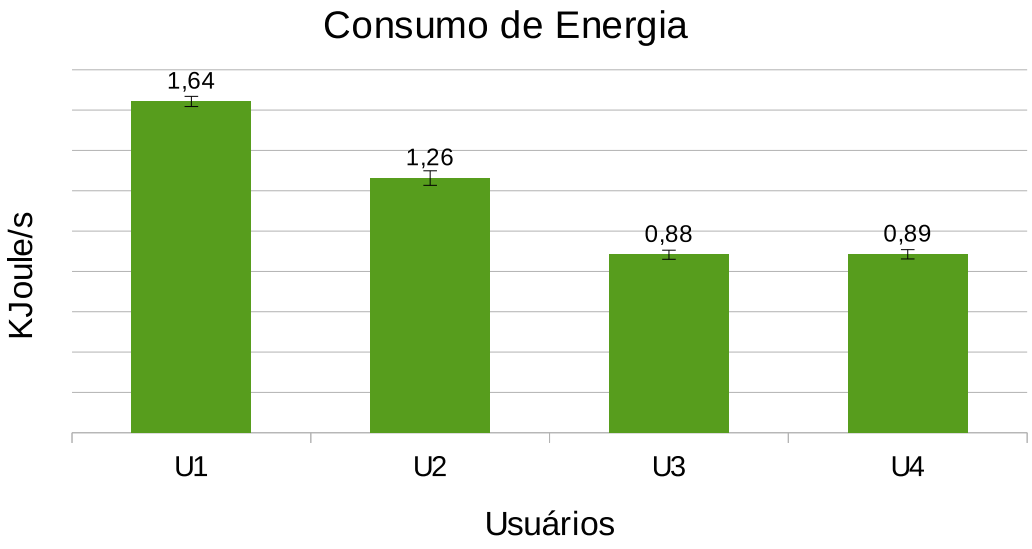

Figura 12 – Consumo dos usuários aplicando HOSEP

Tabela 11 – Comparação de consumo entre HOSEP e EHOSEP na simulação do Modelo 1 sem atrasos

Limite de potência do usuário	U1	U2	U3	U4	Total
80%	-52,50%	-51,20%	-45,47%	-56,98%	-51,67%
100%	-35,58%	-38,24%	-17,37%	-46,61%	-34,94%
150%	-11,61%	-7,33%	-3,77%	-26,15%	-11,73%
Ilimitado	-6,54%	+2,89%	+10,93%	+10,62%	+2,55%

Fonte: do autor

negativos indicam que o EHOSEP foi mais rápido.

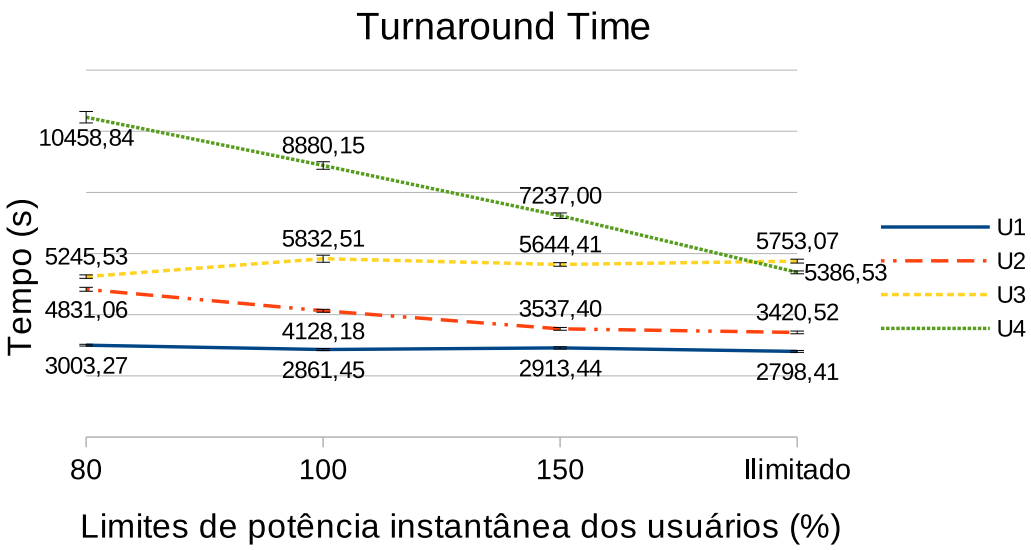

Figura 13 – *Turnaround time* médio dos usuários com todos aplicando limites, utilizando EHOSEP

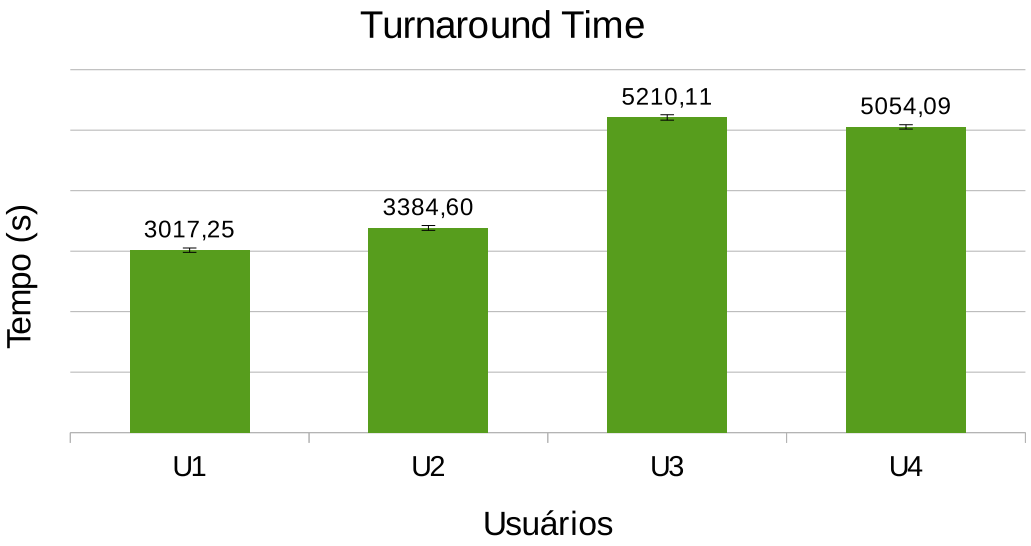

Figura 14 – *Turnaround time* médio dos usuários aplicando HOSEP

Verifica-se pela tabela e pelos gráficos de consumo que quanto mais os usuários limitam suas potências instantâneas de energia mais o consumo do sistema cai, sendo a maior queda de 51,67% nos testes em que os usuários limitam suas potências instantâneas em 80%. Embora interessante, a economia de energia tem como consequência o aumento no *turnaround time* médio dos usuários, sendo U2 e U4 os usuários mais afetados.

Tabela 12 – Comparação de *Turnaround time* entre HOSEP e EHOSEP na simulação do Modelo 1 sem atrasos

Limite de potência dos usuários	U1	U2	U3	U4
80%	-0,46%	+42,73%	+0,68%	+106,93%
100%	-5,16%	+21,97%	+11,94%	+75,70%
150%	-3,44%	+4,51%	+8,33%	+43,19%
Ilimitado	-7,25%	+1,61%	+10,42%	+6,57%

Fonte: do autor

Para o limite de 80% U2 apresenta um aumento de 42,73% no *turnaround time* médio, enquanto U4 tem um aumento de 106,93%. Tais aumentos decorrem do fato de a redução de potência instantânea desses usuários ter um impacto maior no poder computacional a que eles tem acesso. Como as máquinas desses usuários são muito eficientes energeticamente, a potência total de suas quotas é baixa, fazendo com que a porcentagem de 80% reflita em um valor absoluto muito menor do que ocorre para U1 e U3.

No entanto é importante salientar que, à exceção de U4, a economia de energia é sempre superior ao aumento no *turnaround time* médio, ocorrendo até mesmo o caso de U1, que tem sempre uma redução no *turnaround time* médio. Também é importante notar que este favorecimento de U1 acarreta prejuízo para os outros usuários, especialmente para U3. Tal prejuízo é observável no momento em que não há limitação de potência, em que também é perceptível que U2 é favorecido sobre U3 e U4, e que U4 é favorecido sobre U3, devido à melhor eficiência energética dos recursos compartilhados por U2 e U4.

Nota-se também pelos resultados que quando os usuários deixam suas potências instantâneas ilimitadas o EHOSEP consome 2,55% mais energia que o HOSEP. Isto ocorre devido ao favorecimento do usuário U1, que retém máquinas eficientes energeticamente, apresentando como resultado um *turnaround time* 7,25% menor nos testes do algoritmo EHOSEP e um consumo de energia 6,54% menor. Era esperado que o EHOSEP favorecesse mais U2 devido à eficiência energética de suas máquinas.

Entretanto, os bons resultados de U1 não anulam os esforços do algoritmo já que a eficiência das máquinas de U1 não é baixa, e o poder computacional delas é os mais alto detre os nós da grade. Logo, é interessante aos propósitos do EHOSEP que U1 seja favorecido, pois a participação de usuários como ele no sistema contribuirá significativamente para poder computacional da grade com impacto amortizado no consumo de energia.

4.4.2 Teste atrasando U2

Para este teste, simulou-se o Modelo 1 com o atraso da submissão das tarefas de U2, que é o usuário que compartilha as máquinas mais eficientes da grade. Nos respectivos casos de teste os limites de potência de U2 foram 80%, 100%, 150% e ilimitado, enquanto

os demais usuários não limitaram suas potências instantâneas em todos os casos de teste. Todos os casos de teste tiveram 60 simulações e os resultados apresentados são as médias do que foi calculado nos 60 testes, apresentando-se também o intervalo de confiança para 95% de certeza.

Na Figura 15 é apresentado o gráfico de consumo dos usuários nos testes com EHOSEP, enquanto na Figura 16 o gráfico apresenta o consumo nos testes com o HOSEP. Da mesma forma as Figuras 17 e 18 apresentam os gráficos de *turnaround time* médio para os testes com EHOSEP e HOSEP respectivamente. A relação dos resultados obtidos com a simulação do EHOSEP para com os resultados da simulação do HOSEP podem ser vistos nas Tabelas 13 e 14.

Pelos resultados obtidos nota-se que quanto mais U2 restringe sua potência instantânea maior fica seu *turnaround time* médio, que para o limite de 80% fica 44,88% maior do que seria na aplicação do HOSEP. Conforme a restrição de potência instantânea diminui o *turnaround time* médio de U2 melhora mas apresenta redução em relação ao resultado do HOSEP apenas quando a potência instantânea é ilimitada, sugerindo inicialmente que o EHOSEP não está favorecendo U2 pela redução de potência.

É importante notar, entretanto, que há uma grande diferença entre o *turnaround time* médio para o caso de limite 150% e para o caso de potência instantânea ilimitada, mostrando que U2 pode aplicar outros limites e obter redução no *turnaround time* médio. Tal cenário é explicado pelo baixa potência dos recursos de U2, que faz com que os limites de potência aplicados nos testes provoquem uma restrição severa no poder computacional que atende U2, afetando seu *turnaround time* médio.

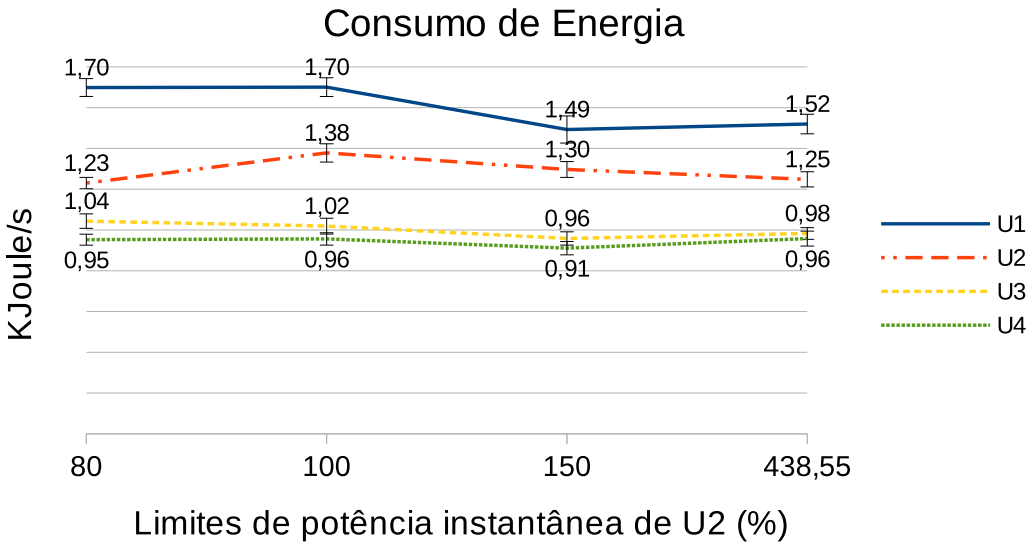

Figura 15 – Consumo dos usuários (S2M1SC) com potência instantânea de U2 limitada e dos demais ilimitada, aplicando EHOSEP

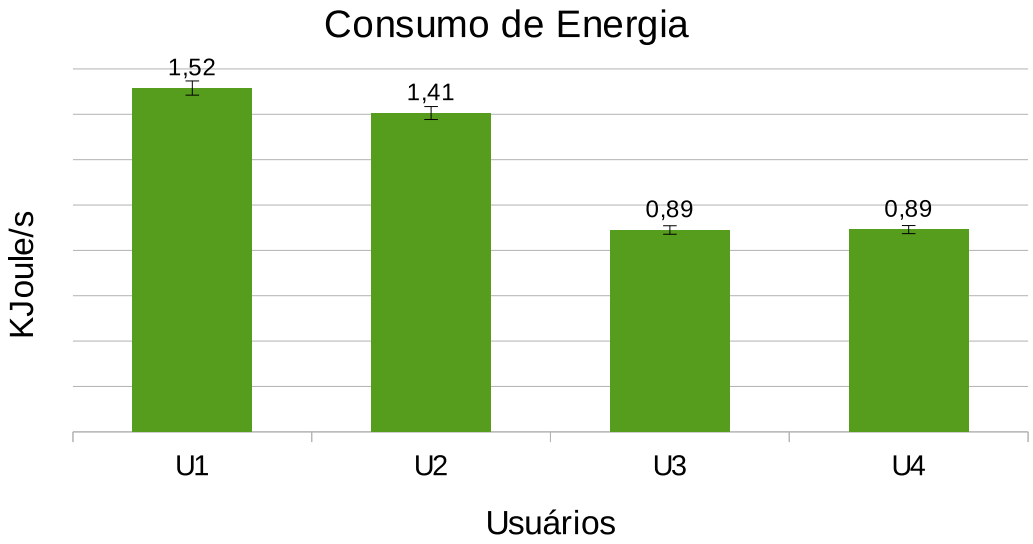

Figura 16 – Consumo dos usuários (S2M1SC), aplicando HOSEP

Tabela 13 – Comparação de consumo entre HOSEP e EHOSEP na simulação do Modelo 1 com atraso de U2

Limite de potência de U2	U1	U2	U3	U4	Total
80%	+12,08%	-12.50%	+17,24%	+6,80%	+4,72%
100%	+12,22%	-1,95%	+14,50%	+7,21%	+7,46%
150%	-1.50%	-7,73%	+7,68%	+2,13%	-0,93%
Ilimitado	+0,27%	-11,18%	+10,43%	+7,40%	+0,12%

Fonte: do autor

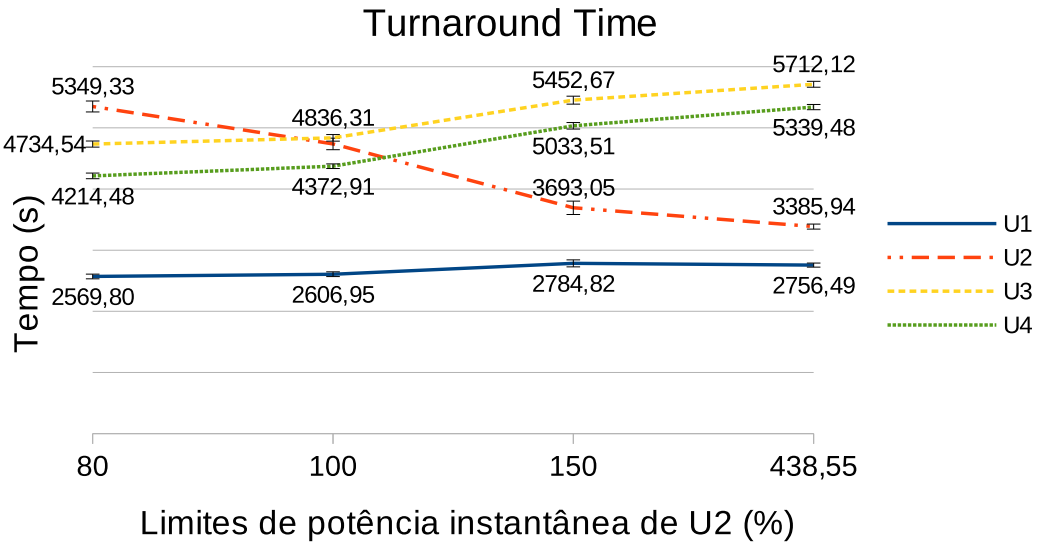

Figura 17 – *Turnaround time* médio dos usuários (S2M1SC) com potência instantânea de U2 limitada e dos demais ilimitada, aplicando EHOSEP

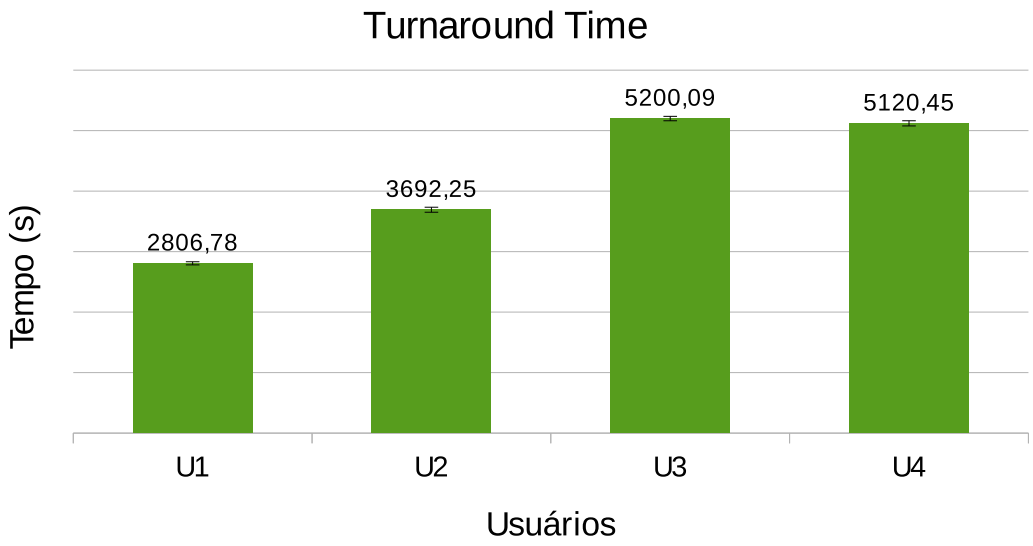

Figura 18 – *Turnaround time* médio dos usuários (S2M1SC), aplicando HOSEP

Tabela 14 – Comparação de *Turnaround time* entre HOSEP e EHOSEP na simulação do Modelo 1 com atraso de U2

Limite de potência de U2	U1	U2	U3	U4
80%	-8,44%	+44,88%	-8,95%	-17,69%
100%	-7,11%	+28,29%	-7,00%	-14,59%
150%	-0,78%	+0,02%	+4,85%	-1,70%
Ilimitado	-1,79%	-8,29%	+9,84%	+4,27%

Fonte: do autor

4.5 Avaliação da estratégia Best Fit de alocação de tarefas

Ao alocar tarefas para máquinas livres, o algoritmo EHOSEP escolhe sempre a tarefa em espera que apresentar o menor tamanho, enquanto a máquina livre escolhida é aquela para a qual o consumo previsto para a execução da tarefa seja o menor. Dessa forma, o EHOSEP emprega uma estratégia *Worst Fit*.

Utilizar a estratégia oposta (*Best Fit*), alocando as tarefas de maior tamanho primeiramente poderia garantir a escolha das máquinas mais eficientes para tais tarefas, reduzindo o consumo geral de energia, já que com *Worst Fit* estas tarefas maiores podem ser forçadas a executar em máquinas de alta potência. Dessa forma as tarefas de maior duração levariam a um grande consumo de energia.

Para verificar se de fato a estratégia *Best Fit* realmente levaria a uma economia de energia, e também verificar como o *turnaround time* médio dos usuários seria afetado, foram feitos os mesmos testes da seção anterior, desta vez comparando as estratégias de alocação de tarefas.

Na tabela 15 é possível verificar como o consumo de energia dos usuários, no teste sem atraso, variou nos resultados obtidos com *Best Fit*, em relação aos resultados obtidos com *Worst Fit*. Já a tabela 16 apresenta a variação do *turnaround time* para os mesmos testes.

Tabela 15 – Comparação de consumo de energia entre *Worst fit* e *Best fit* na simulação do Modelo 1 sem atrasos

Limite de potência dos usuários	U1	U2	U3	U4
80%	+7,60%	+2,29%	-0,38%	+3,81%
100%	+5,60%	+2,71%	+17,27%	+5,85%
150%	+23,18%	+7,71%	-6,12%	+7,54%
Ilimitado	+9,87%	+4,70%	+9,49%	+9,29%

Fonte: do autor

Tabela 16 – Comparação de *Turnaround time* entre *Worst fit* e *Best fit* na simulação do Modelo 1 sem atrasos

Limite de potência dos usuários	U1	U2	U3	U4
80%	+29,34%	+30,45%	+30,81%	+43,01%
100%	+22,67%	+29,68%	+39,98%	+42,16%
150%	+25,77%	+22,87%	+18,77%	+34,51%
Ilimitado	+21,84%	+19,31%	+20,71%	+21,14%

Fonte: do autor

É possível notar que no teste sem atraso de submissão de tarefas a estratégia *Best Fit* levou a um aumento significativo do consumo de energia em quase todos os testes, assim como do valor de *turnaround time*. O mesmo comportamento foi observado com os resultados dos testes feitos atrasando a submissão de tarefas de U2, que podem ser observados nas tabelas 17 (consumo) e 18 (*turnaround time*).

Tabela 17 – Comparação de consumo de energia entre *Worst fit* e *Best fit* na simulação do Modelo 1 com atraso de U2

Limite de potência de U2	U1	U2	U3	U4
80%	+8,45%	+8,26%	+6,93%	+0,017%
100%	+7,18%	+0,56%	+6,19%	+1,88%
150%	+20,27%	+13,68%	+13,07%	+13,15%
Ilimitado	+12,11%	+3,21%	+8,02%	+14,23%

Fonte: do autor

Os resultados obtidos indicam que a estratégia *Best Fit* não é adequada para o modelo de sistema avaliado, isto é, sistemas em que existem tarefas com cargas variadas e a escolha por máquinas leva em consideração sua eficiência energética. Dessa forma, ao priorizar as tarefas de maior tamanho, no conjunto de tarefas em espera, o algoritmo

Tabela 18 – Comparação de *Turnaround time* entre *Worst fit* e *Best fit* na simulação do Modelo 1 com atraso de U2

Limite de potência de U2	U1	U2	U3	U4
80%	+16,97%	+41,94%	+20,49%	+11,58%
100%	+18,97%	+33,30%	+22,98%	+16,74%
150%	+19,52%	+26,56%	+22,28%	+20,39%
Ilimitado	+18,67%	+25,07%	+20,34%	+20,86%

Fonte: do autor

deixa tarefas consideradas médias e pequenas executando em máquinas de baixa eficiência energética. Como estas tarefas são mais numerosas, o consumo de energia do sistema aumenta com a estratégia *Best Fit*. Também ocorreu aumento no *turnaround time* médio, pois executar primeiro as tarefas maiores levou a um aumento no *turnaround time* de tarefas menores, que ficam mais tempo em espera com a estratégia *Best Fit*, e são mais numerosas.

4.6 Comparação entre compartilhar e não compartilhar

Além de avaliar a eficiência coletiva da aplicação do EHOSEP, o que foi feito até agora, é necessário verificar também sua eficiência individual. Isto implica em verificar se para um dado usuário é vantajoso participar da grade ou manter suas máquinas privadamente. Para tanto é preciso verificar como o consumo de energia das máquinas de um usuário (consumo local), e o *turnaround time* de suas tarefas, se compara com aqueles obtidos quando participam da grade.

Para este teste foram gerados aleatoriamente 60 *traces* de tarefas em quantidades, e de tamanhos conforme já definido anteriormente para a simulação do Modelo 1, sem atraso de nenhum usuário e sem *checkpointing* de tarefas. A partir desses *traces* foram extraídos *traces* contendo apenas as tarefas de cada usuário.

Foram então simulados os *traces* completos com o Modelo 1 aplicando EHOSEP sem atrasar os usuários e com todos eles aplicando os mesmos limites (80%,100%,150% e ilimitado), da mesma maneira que os testes das duas seções anteriores. Os *traces* extraídos para usuários isolados foram simulados com modelos em que cada um dos usuários está só, executando apenas suas tarefas em seus recursos particulares. Para estes modelos o escalonador aplicado foi o *Shortest Job First* (SJF), que escalona as menores tarefas para a máquina livre de maior poder computacional. O comportamento do SJF é equivalente ao do EHOSEP quando se considera apenas um usuário por vez.

A partir das simulações foram calculados o consumo local em KJ/s e o *turaround time* médio, e foi calculada a variação relativa destas métricas, entre o que foi obtido nos testes com EHOSEP e o que foi calculado nos testes de usuários isolados (SJF). No

gráfico da Figura 19 estão dispostos os resultados obtidos na forma de pontos. No eixo das abscissas está representada a variação relativa do consumo local, com valores negativos indicando que o consumo foi menor ao se compartilhar as máquinas na grade. Já no eixo das ordenadas está representada a variação relativa do turnaround time médio, com valores negativos indicando que tempo foi menor com compartilhamento. Assim, um ponto com valor negativo indica que naquele caso, para aquele parâmetro, foi melhor compartilhar as máquinas na grade do que fazer uso privativo delas.

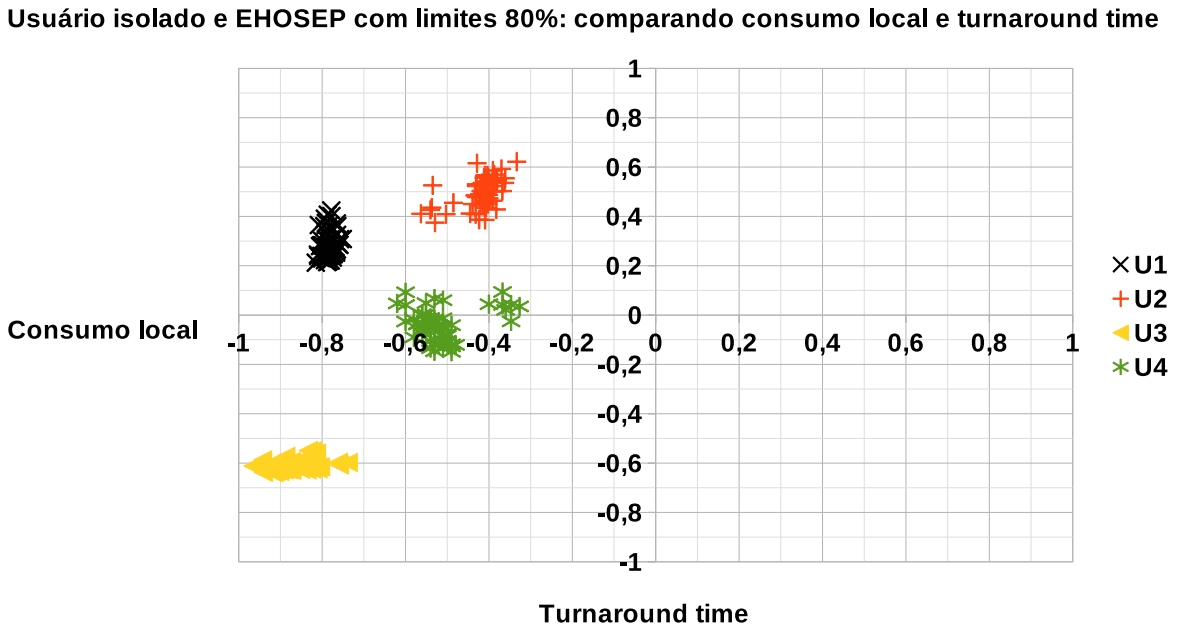

Figura 19 – Comparação entre consumo local e *turnaround time* com usuários isolados (SJF) e participando de uma grade cooperativa (EHOSEP, limites 80%)

É possível observar que todos os usuários obtiveram grande economia de energia ao participarem da grade com limites de potência de 80%, com o usuário U1 e o usuário U3 obtendo as maiores economias em relação ao que teriam caso não participassem da grade. Com relação ao *turnaround time* médio, os usuários U1 e U2 tiveram um aumento de 20% a 60% no tempo de execução, dependendo do caso, devido à disputa por recursos e também ao fato de as máquinas de U3 e U4 provocarem um aumento no tempo de execução de eventuais tarefas de U1 e U2, com relação ao que teriam executando sozinhos.

Já para U3 o *turnaround time* médio na grade foi 60% menor devido ao uso que fez de máquinas muito mais poderosas do que as que ele possui. O mesmo não ocorreu para U4 devido ao limite de potência, pois como suas máquinas consomem pouco, o poder computacional disponível para ele fica limitado, não permitindo a ele obter menor tempo de execução das tarefas.

Na Figura 20 podem ser vistos os resultados obtidos comparando as métricas dos testes isolados com o teste EHOSEP com limites de potência em 100%. Com relação

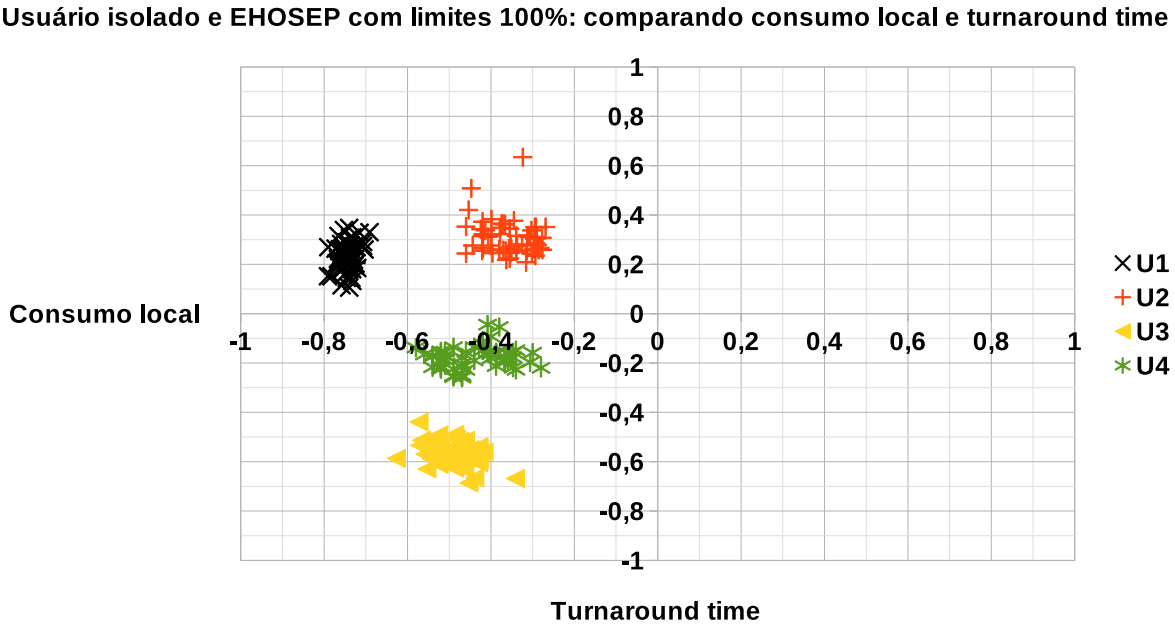

Figura 20 – Comparação entre consumo local e *turnaround time* com usuários isolados (SJF) e participando de uma grade cooperativa (EHOSEP, limites 100%)

ao resultado anterior, a economia de energia de todos os usuários diminuiu um pouco, com exceção de U3 que viu sua economia de consumo local cair de quase 100% para um intervalo de 60% a 40%.

Com limite em 100% o *turnaround time* de U1 e U2 apresentou aumento menor em relação ao crescimento visto no caso anterior. Isso ocorre pois o novo limite permite a alocação de máquinas de maior poder computacional. O mesmo ocorreu com U4, que passou a apresentar ligeira redução no tempo de execução das tarefas.

Nas Figuras 21 e 22 estão os resultados com limite de potência de 150% e com potência ilimitada. Nota-se por estes gráficos, e pelos outros também, que conforme os limites aumentam, a economia de consumo local cai e os *turnaround times* caem mais para U3 e U4, e aumentam menos (até 30% no teste sem limites) para U1 e U2. É possível notar também que U3 apresenta alguns resultados com aumento de consumo local de até 10%.

Com estes resultados, fica comprovado que a principal vantagem de se participar de uma grade colaborativa com escalonador EHOSEP é a redução no consumo local, reduzindo o custo financeiro com o uso das máquinas. Para usuários com poder computacional relevante e máquinas eficientes (U1, U2 e U4) a economia de energia é de pelo menos 20%, conforme pode ser visto nos resultados dos teste com potência instantânea ilimitada.

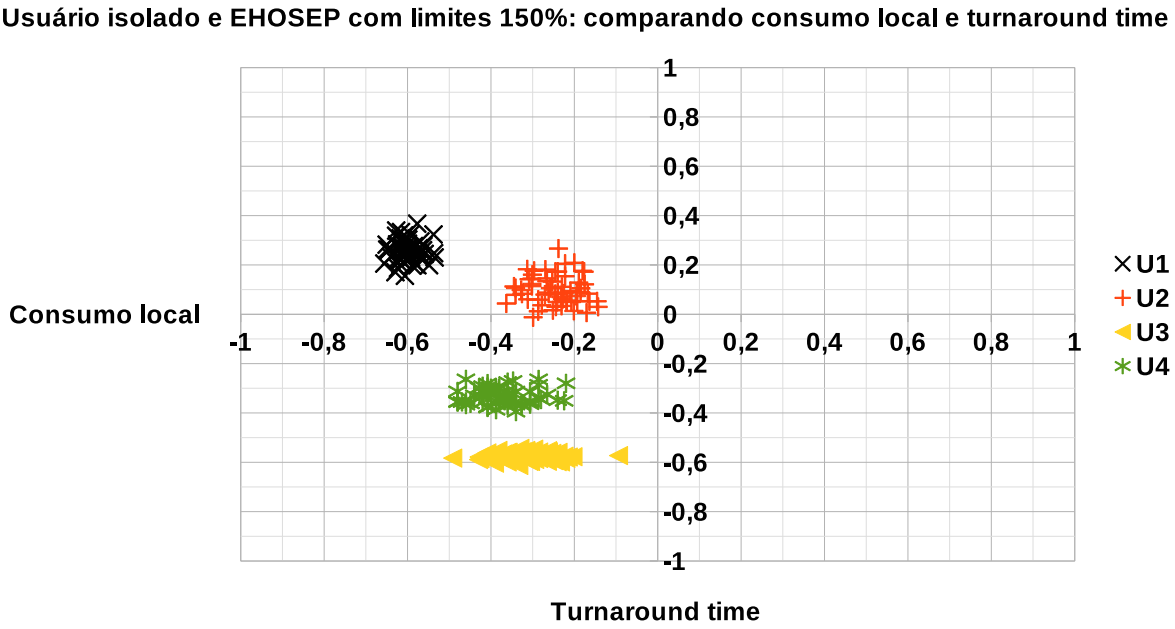

Figura 21 – Comparação entre consumo local e *turnaround time* com usuários isolados (SJF) e participando de uma grade cooperativa (EHOSEP, limites 150%)

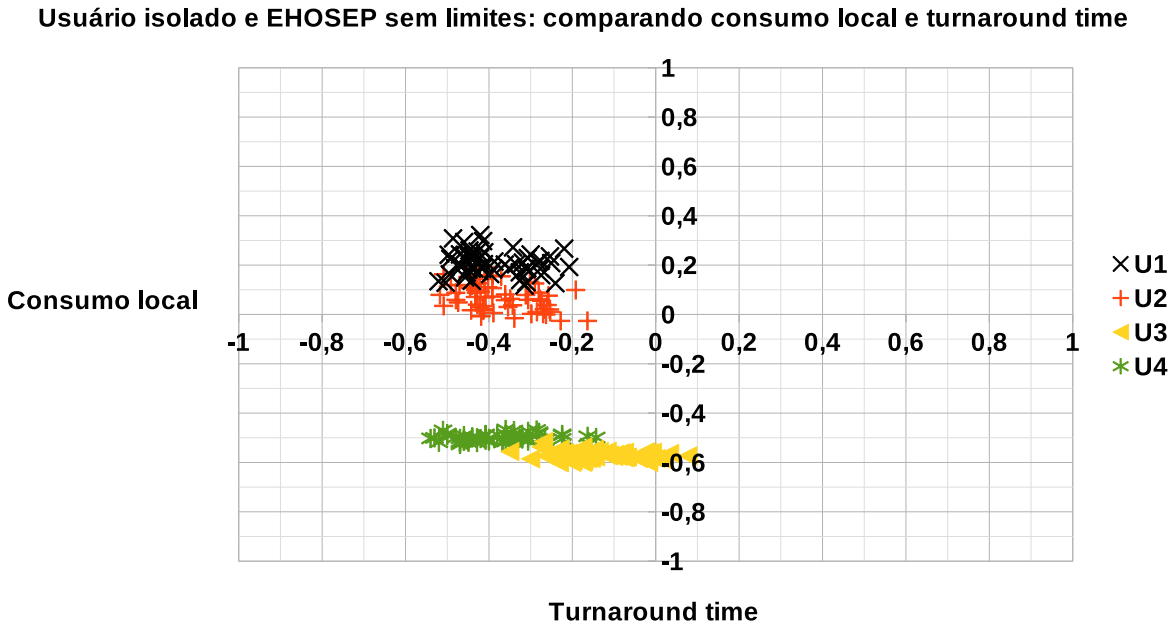

Figura 22 – Comparação entre consumo local e *turnaround time* com usuários isolados (SJF) e participando de uma grade cooperativa (EHOSEP, sem limites)

4.7 Análise geral dos resultados

Foi possível observar que o EHOSEP mantém as máquinas da grade ocupadas, sempre que houver demanda para tanto. Em particular, uma consequência aparentemente contraditória disso é que se um usuário restringe seu uso em função de um limite de potência muito baixo, os usuários do sistema que tenham maior limite acabam utilizando as máquinas que ficariam ociosas. Isso é verificável pois nessa situação a satisfação desses usuários acaba aumentando, como consequência do teste da linha 7 do Algoritmo 3, que impede um usuário assumir mais máquinas do que seu limite de potência permite. Caso todos os usuários coloquem limites altos, o algoritmo se reduz ao poder computacional ofertado por cada um.

Observou-se também consistência entre os resultados no momento em que todos os usuários apresentam o mesmo limite de potência absoluto, no último caso de teste da cada sequência de testes. Isso confirma em escala geral que o EHOSEP atende à hipótese de teste I), mantendo a proporcionalidade da satisfação com relação ao poder computacional oferecido pelos usuários.

Os testes com o primeiro modelo deixaram muito clara a vantagem dos usuários que oferecem recursos eficientes. Neste modelo U2 e U4 são os usuários mais eficientes energeticamente e, quando limitam suas potências instantâneas em 150% nos testes em que são atrasados, chegam a ficar mais satisfeitos que U1, que é o usuário que oferece o maior poder computacional. Por outro lado, a vantagem dos usuários que limitam a potência instantânea também é observável, especialmente no teste S3M1SC, em que U3 alcança satisfação maior que todos, ficando mais satisfeito quanto mais limita sua potência instantânea, mesmo sendo o usuário com os recursos computacionais menos eficientes energeticamente.

Os testes com o segundo modelo mostraram que quando as quotas dos usuários apresentam poder computacional total similares, a satisfação passa a ser proporcional à eficiência energética e ao poder individual das máquinas de cada usuário. Nestes testes U1 fica sempre mais satisfeito que U2, mesmo tendo este os recursos mais eficientes energeticamente. Isso ocorre porque as máquinas de U1 apresentam maior poder computacional individualmente.

Ainda nos testes com o segundo modelo foi possível verificar que usuários que compartilham recursos de baixo poder computacional e pouco eficientes ficam pouco satisfeitos, independentemente do limite de potência que apliquem. O usuário que mais ilustra este caso é U3, que em todos os testes do Modelo 2 apresenta os menores valores de satisfação dentre os usuários.

Além da satisfação dos usuários também se avaliou o *turnaround time* médio e o consumo de energia total e local de cada usuário em diferentes cenários. Os resultados

destes testes permitiram observar que o EHOSEP obteve redução nos consumos, quando comparado ao HOSEP e ao SJF para usuários isolados, apresentando aumentos no *turnaround time* médio. Apesar destes aumentos, a economia de energia foi significativa, especialmente quando se compara o EHOSEP com o SJF de usuários isolados, confirmando a validade da proposta do EHOSEP e o cumprimento de seus objetivos. Ainda com relação aos testes de consumo e *turnaround time*, mostrou-se que a estratégia *Worst Fit* de alocação de tarefas obteve melhores resultados que a estratégia *Best Fit* para a carga de trabalho simulada.

4.8 Considerações Finais

Neste capítulo se descreveu a metologia de teste, os modelos simulados e todos os outros procedimentos aplicados para avaliar o escalonador EHOSEP. Definiu-se ainda um conjunto de hipóteses, derivadas dos objetivos do EHOSEP, para delimitar como os testes deveriam ser planejados.

As hipóteses foram confirmadas na maior parte dos testes. Em particular, a hipótese relacionada à justiça de propriedade se confirmou em 100% dos casos, o que sustenta a afirmação de que o algoritmo EHOSEP garante a justiça de propriedade de recursos em grades cooperativas, favorecendo ainda os usuários que limitam a potência instantânea, e que oferecem recursos de melhor eficiência energética. Entretanto, nota-se também pelos resultados dos testes, especialmente os da sequência S3M2SC, que o EHOSEP também dá importância ao poder computacional individual das máquinas, o que não havia sido considerado.

Dessa forma o EHOSEP cumpre seu objetivo de fomentar a formação de grades cooperativas por usuários que contribuam significativamente com o sistema, ou seja, com recursos eficientes energeticamente e volume de poder computacional significativo.

5 Conclusões e Perspectivas

Nesta dissertação apresentou-se o algoritmo EHOSEP para escalonamento de tarefas em grades cooperativas. Ao longo dos Capítulos 3 e 4 foram apresentadas sua especificação, implementação e avaliação por meio de testes por simulação. Neste capítulo são apresentadas as conclusões do trabalho, suas contribuições e seus problemas. Também são apontados alguns possíveis desdobramentos do projeto.

5.1 Conclusões

Um aspecto importante em grades cooperativas é garantir aos seus usuários o recebimento de poder computacional equivalente ao por eles fornecidos sempre que houver demanda. Entretanto, esse não deve ser o único aspecto a ser considerado. O consumo de energia, que é um fator primordial tanto para grades cooperativas quanto sistemas computacionais de maneira geral, precisa ser tratado para evitar aumento de custos. Assim, participar de uma grade cooperativa implica não apenas em afetar o poder computacional dela mas também seu consumo de energia ou eficiência energética.

Partindo do HOSEP, que trata apenas de justiça de propriedade, o EHOSEP tem por objetivo incentivar a formação de grades cooperativas mais eficientes energeticamente e a diminuição no consumo de energia pelos usuários. Para tanto se busca favorecer usuários que ofertem máquinas eficientes e/ou que limitem a potência instantânea que consomem. Esse favorecimento pode ser obtido no momento de interromper uma tarefa, para garantir justiça de propriedade, ao se evitar escolher tarefas de usuários que estejam limitando o quanto podem consumir de energia.

As simulações mostraram que os objetivos do projeto foram atingidos. O EHOSEP favorece usuários com recursos mais eficientes, e usuários que limitam sua potência instantânea ficam mais satisfeitos. Um corolário percebido é que os usuários oferecendo maior poder computacional são também beneficiados pela observação da quota de cada participante. Os resultados também evidenciaram uma economia na energia consumida nas quotas de cada usuário quando estes participam da grade. Dessa forma dá-se legítimo estímulo à criação e à expansão de grades cooperativas, gerando sistemas mais poderosos e eficientes energeticamente.

5.1.1 Contribuições

Como visto ao longo do Capítulo 3, as métricas de decisão do EHOSEP, que são o *Coeficiente Sistema/Quota*, *Potência Instantânea*, *Diferencial de Poder* e *Poder*

Remanescente Esperado, são simples de calcular e não exigem dados cuja obtenção requeira equipamentos de medição específicos. Com isso a operação do escalonador não exige equipamentos específicos de medição, nem que as máquinas do sistema ofereçam funcionalidades específicas como o DVFS. Dessa forma a grade formada pode ter a participação de usuários com *hardware* tecnologicamente defasado e/ou mais barato, embora isso resulte em uma redução de sua prioridade.

O EHOSEP pode ser implementado com estruturas de dados simples, como listas e vetores. Considerando que grades cooperativas dificilmente têm mais que algumas dezenas de usuários (fornecedores de recursos) e que tais usuários não devem fornecer grandes quantidades de máquinas, a ordenação dos dados nas estruturas mencionadas não exige um método específico. Além disso, processadores modernos são capazes de ordenar listas e vetores com dezenas de milhares de elementos em períodos de tempo cuja ordem é de milissegundos, senão microssegundos (57). Dessa forma a ordenação levaria muito menos tempo que os períodos entre as chamadas do escalonador, que ocorrem a cada dezenas de segundos. Estas características fazem do EHOSEP um algoritmo escalável para grades cooperativas.

Apesar de projetado para grades cooperativas, o EHOSEP trata noções genéricas de propriedade de máquinas e consumo de energia. Dessa forma o uso do escalonador em sistemas de computação em nuvem cooperativos pode ser feito com facilidade. Quanto ao tipo de tarefas tratado, a única exigência é que sejam independentes, o que é uma característica bem genérica.

Outro aspecto importante deste trabalho é a métrica de satisfação dos usuários (Equação 4.1). Do ponto de vista de modelagem, obter uma única equação que reúna os aspectos fundamentais da satisfação do usuário em uma grade cooperativa é uma tarefa difícil. Com a equação obtida tem-se uma métrica representativa, capaz de definir a satisfação geral com um cálculo simples e elegante.

5.1.2 Dificuldades encontradas

O principal problema no desenvolvimento desta política de escalonamento foi atender os princípios de justiça de propriedade (garantia de poder computacional), ao mesmo tempo que se busca redução no consumo de energia. A definição dos critérios adotados demandou a proposição de diversas formas de fazer a alocação e preempção de recursos no sistema, buscando minimizar o consequente *slowdown* nas tarefas dos usuários.

Do ponto de vista de testes as dificuldades encontradas foram de dois tipos. Primeiro, foi impossível comparar, por simulação, o EHOSEP com escalonadores que aplicam DVFS. Isso ocorreu porque não houve condições para implementar a simulação de mecanismos de DVFS no iSPD, que foi o simulador utilizado nos testes. Segundo, não foi possível avaliar

o EHOSEP em um ambiente real, uma vez que não foi possível se ter acesso a uma grade real, ainda que não fosse cooperativa.

5.2 Direções futuras

A partir das conclusões apresentadas aqui é possível indicar algumas possíveis direções para a continuidade desse trabalho. As principais delas são:

- Garantia de turnaround time: é preciso investigar se ajustes no critério de justiça podem garantir aos usuários um *turnaround time* médio menor ou igual ao que teriam sem participar da grade, e o que isso implicaria no consumo de energia. Uma consequência dessa investigação seria a proposição de alternativas de uso, buscando maior economia ou maior velocidade.

- Descentralização do EHOSEP: como é sabido, algoritmos centralizados como o EHOSEP são mais suscetíveis a falhas. Assim, a sua descentralização, possivelmente com a divisão da grade em grupos gerenciados por escalonadores locais EHOSEP, além de um meta-escalonador global, pode ser interessante ao se trabalhar em dois níveis de decisão.

- Comparação com algoritmos baseados em DVFS: embora a não necessidade de mecanismos específicos seja um aspecto favorável do EHOSEP, ainda é preciso que se avalie se a redução de consumo por ele obtida é comparável com a obtida por mecanismos como o DVFS. Nesse sentido se pretende inicialmente comparar o comportamento do MOSP-Energy (51) e do EHOSEP em situações semelhantes, uma vez que os dois tratam grades cooperativas.

- Testes em ambiente real: buscando confirmar os resultados obtidos via simulação por meio da realização de testes em grades cooperativas reais.

- Escalonamento de tarefas com dependência: apesar de tarefas independentes formarem um conjunto bastante amplo das aplicações de alto desempenho, tratar tarefas dependentes, como as de *workflow* é sempre interessante. Assim, é razoável fazer com que o EHOSEP passe a escalonar tarefas com dependência, possivelmente tratando diferentemente o consumo de energia nas múltiplas etapas de cada tarefa.

- Escalonamento em sistemas com pagamento pelo consumo: rever os critérios de justiça e consumo para tratar sistemas cooperativos em que os participantes paguem pelo consumo de energia que suas tarefas gerem em máquinas de outros usuários. A adição de cobrança por consumo de energia permite uma melhor gestão de consumo e, de certo modo, a aplicação do EHOSEP para outros tipos de grades computacionais, incluíndo computação em nuvem.

Referências

1 BELOGLAZOV, A.; BUYYA, R.; LEE, Y. C.; ZOMAYA, A. et al. A taxonomy and survey of energy-efficient data centers and cloud computing systems. *Advances in computers*, Academic Press, v. 82, n. 2, p. 47–111, 2011.

2 FALAVINHA, J.; MANACERO, A.; LIVNY, M.; BRADLEY, D. The owner share scheduler for a distributed system. In: INTL CONF ON PARALLEL PROCESSING WORKSHOPS, ICPPW'09. Vienna, Austria, 2009. p. 298–305.

3 FORTE, C.; MANACERO, A.; LOBATO, R.; SPOLON, R.; FALAVINHA-JR, J. Política de escalonamento owner-share para sistemas de grades heterogeneas. In: ANAIS DO XVII SIMP. EM SIST. COMPUTACIONAIS DE ALTO DESEMPENHO - WSCAD. Aracaju, Brasil, 2016. p. 206–217.

4 IOSUP, A.; EPEMA, D. Grid computing workloads. *IEEE Internet Computing*, v. 15, n. 2, p. 19–26, March 2011. ISSN 1089-7801.

5 TANENBAUM, A. S.; STEEN, M. V. *Sistemas Distribuídos Princípios e Paradigmas*. São Paulo: Editora Pearson Prentice Hall, 2007.

6 FOSTER, I.; KESSELMAN, C. *The Grid 2: Blueprint for a new computing infrastructure*. San Francisco, CA, EUA: Elsevier, 2003.

7 ABBAS, A. *GRID COMPUTING: A Practical Guide to Technology and Applications*. Rockland, MA, USA: Charles River Media, Inc., 2003. ISBN 1584502762.

8 XHAFA, F.; ABRAHAM, A. Computational models and heuristic methods for grid scheduling problems. *Future Generation Computer Systems*, v. 26, n. 4, p. 608 – 621, 2010. ISSN 0167-739X. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0167739X09001782>.

9 HENRY, G. J. The unix system: The fair share scheduler. *AT&T Bell Laboratories Technical Journal*, Blackwell Publishing Ltd, v. 63, n. 8, p. 1845–1857, 1984. ISSN 1538-7305. Disponível em: <http://dx.doi.org/10.1002/j.1538-7305.1984.tb00068.x>.

10 KOŁODZIEJ, J.; KHAN, S. U.; WANG, L.; KISIEL-DOROHINICKI, M.; MADANI, S. A.; NIEWIADOMSKA-SZYNKIEWICZ, E.; ZOMAYA, A. Y.; XU, C.-Z. Security, energy, and performance-aware resource allocation mechanisms for computational grids. *Future Generation Computer Systems*, Elsevier, v. 31, p. 77–92, 2014.

11 ZHANG, L.; LI, K.; XU, Y.; MEI, J.; ZHANG, F.; LI, K. Maximizing reliability with energy conservation for parallel task scheduling in a heterogeneous cluster. *Information Sciences*, Elsevier, v. 319, p. 113–131, 2015.

12 LI, K.; TANG, X.; LI, K. Energy-efficient stochastic task scheduling on heterogeneous computing systems. *IEEE Transactions on Parallel and Distributed Systems*, IEEE, v. 25, n. 11, p. 2867–2876, 2014.

13 HOSSEINIMOTLAGH, S.; KHUNJUSH, F. Migration-less energy-aware task scheduling policies in cloud environments. In: 28TH INTL CONF ON ADVANCED INFORMATION NETWORKING AND APPLICATIONS WORKSHOPS (WAINA). *IEEE*. Victoria, BC, Canada, 2014. p. 391–397.

14 LEI, H.; WANG, R.; ZHANG, T.; LIU, Y.; ZHA, Y. A multi-objective co-evolutionary algorithm for energy-efficient scheduling on a green data center. *Computers & Operations Research*, Elsevier, v. 75, p. 103–117, 2016.

15 ZHANG, Y.; WANG, Y.; HU, C. Cloudfreq: Elastic energy-efficient bag-of-tasks scheduling in dvfs-enabled clouds. In: 21ST INTL CONF ON PARALLEL AND DISTRIBUTED SYSTEMS (ICPADS). *IEEE*. Melbourne, Australia, 2015. p. 585–592.

16 ALAHMADI, A.; CHE, D.; KHALEEL, M.; ZHU, M. M.; GHODOUS, P. An innovative energy-aware cloud task scheduling framework. In: 8TH INTL CONF ON CLOUD COMPUTING. *IEEE*. New York, EUA, 2015. p. 493–500.

17 TCHERNYKH, A.; PECERO, J. E.; BARRONDO, A.; SCHAEFFER, E. Adaptive energy efficient scheduling in peer-to-peer desktop grids. *Future Generation Computer Systems*, Elsevier, v. 36, p. 209–220, 2014.

18 ŻOTKIEWICZ, M.; GUZEK, M.; KLIAZOVICH, D.; BOUVRY, P. Minimum dependencies energy-efficient scheduling in data centers. *IEEE Transactions on Parallel and Distributed Systems*, IEEE, v. 27, n. 12, p. 3561–3574, 2016.

19 ZHU, X.; YANG, L. T.; CHEN, H.; WANG, J.; YIN, S.; LIU, X. Real-time tasks oriented energy-aware scheduling in virtualized clouds. *IEEE Transactions on Cloud Computing*, IEEE, v. 2, n. 2, p. 168–180, 2014.

20 COMITO, C.; FALCONE, D.; TALIA, D.; TRUNFIO, P. Energy-aware task allocation for small devices in wireless networks. *Concurrency and Computation: Practice and Experience*, Wiley Online Library, v. 29, n. 1, 2017.

21 GAI, K.; QIU, M.; ZHAO, H.; LIU, M. Energy-aware optimal task assignment for mobile heterogeneous embedded systems in cloud computing. In: INTL CONF ON CYBER SECURITY AND CLOUD COMPUTING (CSCLOUD). *IEEE*. Beijing, China, 2016. p. 198–203.

22 MÄSKER, M.; NAGEL, L.; BRINKMANN, A.; LOTFIFAR, F.; JOHNSON, M. Smart grid-aware scheduling in data centres. *Computer Communications*, Elsevier, v. 96, p. 73–85, 2016.

23 BELDICEANU, N.; FERIS, B. D.; GRAVEY, P.; HASAN, S.; JARD, C.; LEDOUX, T.; LI, Y.; LIME, D.; MADI-WAMBA, G.; MENAUD, J.-M. et al. Towards energy-proportional clouds partially powered by renewable energy. *Computing*, Springer, v. 99, n. 1, p. 3–22, 2017.

24 NIU, Z.; HE, B.; LIU, F. Not all joules are equal: Towards energy-efficient and green-aware data processing frameworks. In: INTL CONF ON CLOUD ENGINEERING (IC2E). *IEEE*. Berlin, Germany, 2016. p. 2–11.

25 ZHAO, Q.; XIONG, C.; YU, C.; ZHANG, C.; ZHAO, X. A new energy-aware task scheduling method for data-intensive applications in the cloud. *Journal of Network and Computer Applications*, Elsevier, v. 59, p. 14–27, 2016.

26 MAQSOOD, T.; TZIRITAS, N.; LOUKOPOULOS, T.; MADANI, S. A.; KHAN, S.; XU, C.-Z. Leveraging on deep memory hierarchies to minimize energy consumption and data access latency on single-chip cloud computers. *IEEE Transactions on Sustainable Computing*, IEEE, 2017.

27 CHATTERJEE, N.; PAUL, S.; MUKHERJEE, P.; CHATTOPADHYAY, S. Deadline and energy aware dynamic task mapping and scheduling for network-on-chip based multi-core platform. *Journal of Systems Architecture*, Elsevier, v. 74, p. 61–77, 2017.

28 DORRONSORO, B.; NESMACHNOW, S.; TAHERI, J.; ZOMAYA, A. Y.; TALBI, E.-G.; BOUVRY, P. A hierarchical approach for energy-efficient scheduling of large workloads in multicore distributed systems. *Sustainable Computing: Informatics and Systems*, Elsevier, v. 4, n. 4, p. 252–261, 2014.

29 TARPLEE, K. M.; MACIEJEWSKI, A. A.; SIEGEL, H. J. Energy-aware profit maximizing scheduling algorithm for heterogeneous computing systems. In: 14TH INTL SYMPOSIUM ON CLUSTER, CLOUD AND GRID COMPUTING (CCGRID). *IEEE*. Chicago, IL, USA, 2014. p. 595–603.

30 MASHAYEKHY, L.; NEJAD, M. M.; GROSU, D.; ZHANG, Q.; SHI, W. Energy-aware scheduling of mapreduce jobs for big data applications. *IEEE transactions on Parallel and distributed systems*, IEEE, v. 26, n. 10, p. 2720–2733, 2015.

31 LI, H.; LI, J.; YAO, W.; NAZARIAN, S.; LIN, X.; WANG, Y. Fast and energy-aware resource provisioning and task scheduling for cloud systems. In: 18TH INTL SYMPOSIUM ON QUALITY ELECTRONIC DESIGN (ISQED). *IEEE*. Santa Clara, CA, USA, 2017. p. 174–179.

32 STAVRINIDES, G. L.; KARATZA, H. D. Simulation-based performance evaluation of an energy-aware heuristic for the scheduling of hpc applications in large-scale distributed systems. In: PROCEEDINGS OF THE 8TH INTL CONF ON PERFORMANCE ENGINEERING COMPANION. *ACM*. L'Aquila, Italy, 2017. p. 49–54.

33 SHI, L.; ZHANG, Z.; ROBERTAZZI, T. Energy-aware scheduling of embarrassingly parallel jobs and resource allocation in cloud. *IEEE Transactions on Parallel and Distributed Systems*, IEEE, v. 28, n. 6, p. 1607–1620, 2017.

34 ITURRIAGA, S.; DORRONSORO, B.; NESMACHNOW, S. Multiobjective evolutionary algorithms for energy and service level scheduling in a federation of distributed datacenters. *International Transactions in Operational Research*, Wiley Online Library, v. 24, n. 1-2, p. 199–228, 2017.

35 RAJU, R.; AMUDHAVEL, J.; KANNAN, N.; MONISHA, M. A bio inspired energy-aware multi objective chiropteran algorithm (eamoca) for hybrid cloud computing environment. In: INTL CONF ON GREEN COMPUTING COMMUNICATION AND ELECTRICAL ENGINEERING (ICGCCEE). *IEEE*. Thuckalay, India, 2014. p. 1–5.

36 TAO, F.; FENG, Y.; ZHANG, L.; LIAO, T. W. Clps-ga: A case library and pareto solution-based hybrid genetic algorithm for energy-aware cloud service scheduling. *Applied Soft Computing*, Elsevier, v. 19, p. 264–279, 2014.

37 AMAR, L.; BARAK, A.; LEVY, E.; OKUN, M. An on-line algorithm for fair-share node allocations in a cluster. In: IEEE. *Cluster Computing and the Grid (CCGrid '07), Proc. of 7th IEEE International Symposium on*. Rio De Janeiro, Brasil, 2007. p. 83 – 91.

38 KAY, J.; LAUDER, P. A fair share scheduler. *Commun. ACM*, ACM, New York, NY, USA, v. 31, n. 1, p. 44–55, jan. 1988. ISSN 0001-0782. Disponível em: <http://doi.acm.org/10.1145/35043.35047>.

39 ANDRADE, N.; BRASILEIRO, F.; CIRNE, W.; MOWBRAY, M. Discouraging free riding in a peer-to-peer cpu-sharing grid. In: 13TH INTL SYMPOSIUM ON HIGH PERFORMANCE DISTRIBUTED COMPUTING. *IEEE*. Honolulu, HI, EUA, 2004. p. 129–137. ISSN 1082-8907.

40 XU, H.; YANG, B. An incentive-based heuristic job scheduling algorithm for utility grids. *Future Generation Computer Systems*, Elsevier, v. 49, p. 1–7, 2015.

41 XIAO, Z.; TONG, Z.; LI, K.; LI, K. Learning non-cooperative game for load balancing under self-interested distributed environment. *Applied Soft Computing*, Elsevier, v. 52, p. 376–386, 2017.

42 YILDIZ, O.; IBRAHIM, S.; ANTONIU, G. Enabling fast failure recovery in shared hadoop clusters: Towards failure-aware scheduling. *Future Generation Computer Systems*, Elsevier, v. 74, p. 208–219, 2017.

43 RUBIO-MONTERO, A.; HUEDO, E.; MAYO-GARCÍA, R. Scheduling multiple virtual environments in cloud federations for distributed calculations. *Future Generation Computer Systems*, Elsevier, v. 74, p. 90–103, 2017.

44 XING, G.; XU, X.; XIANG, H.; XUE, S.; JI, S.; YANG, J. Fair energy-efficient virtual machine scheduling for internet of things applications in cloud environment. *International Journal of Distributed Sensor Networks*, SAGE Publications Sage UK: London, England, v. 13, n. 2, p. 1550147717694890, 2017.

45 VERMA, A.; PEDROSA, L.; KORUPOLU, M.; OPPENHEIMER, D.; TUNE, E.; WILKES, J. Large-scale cluster management at google with borg. In: PROCEEDINGS OF THE TENTH EUROPEAN CONFERENCE ON COMPUTER SYSTEMS. *ACM*. Bordeaux, France, 2015. p. 18.

46 SAXENA, D.; CHAUHAN, R.; KAIT, R. Dynamic fair priority optimization task scheduling algorithm in cloud computing: Concepts and implementations. *International Journal of Computer Network and Information Security*, Modern Education and Computer Science Press, v. 8, n. 2, p. 41, 2016.

47 XIE, G.; LIU, L.; YANG, L.; LI, R. Scheduling trade-off of dynamic multiple parallel workflows on heterogeneous distributed computing systems. *Concurrency and Computation: Practice and Experience*, Wiley Online Library, v. 29, n. 2, 2017.

48 WEN, C.; HE, J.; ZHANG, J.; LONG, X. Pcfs: Power credit based fair scheduler under dvfs for muliticore virtualization platform. In: PROCEEDINGS OF THE INTL CONF ON GREEN COMPUTING AND COMMUNICATIONS & INTL CONF ON CYBER, PHYSICAL AND SOCIAL COMPUTING. *IEEE/ACM*. Hangzhou, China, 2010. p. 163–170.

49 GEORGIOU, Y.; GLESSER, D.; RZADCA, K.; TRYSTRAM, D. A scheduler-level incentive mechanism for energy efficiency in hpc. In: 15TH INTL SYMPOSIUM ON CLUSTER, CLOUD AND GRID COMPUTING (CCGRID). *IEEE/ACM*. Shenzhen, China, 2015. p. 617–626.

50 KIM, N.; CHO, J.; SEO, E. Energy-based accounting and scheduling of virtual machines in a cloud system. In: INTL CONFERENCE ON GREEN COMPUTING AND COMMUNICATIONS (GREENCOM). *IEEE/ACM*. Sichuan, China, 2011. p. 176–181.

51 COHEN, J.; CORDEIRO, D.; RAPHAEL, P. L. F. Energy-aware multi-organization scheduling problem. In: EURO-PAR. *Computer Science Department, Faculty of Sciences, University of Porto*. Porto, Portugal, 2014. p. 186–197.

52 KHEMKA, B.; FRIESE, R.; PASRICHA, S.; MACIEJEWSKI, A. A.; SIEGEL, H. J.; KOENIG, G. A.; POWERS, S.; HILTON, M.; RAMBHAROS, R.; POOLE, S. Utility driven dynamic resource management in an oversubscribed energy-constrained heterogeneous system. In: INTL PARALLEL & DISTRIBUTED PROCESSING SYMPOSIUM WORKSHOPS (IPDPSW). *IEEE*. Phoenix, AZ, EUA, 2014. p. 58–67.

53 MENEZES, D.; MANACERO, A.; LOBATO, R.; SILVA, D. da; SPOLON, R. Scheduler simulation using ispd, an iconic-based computer grid simulator. In: IEEE SYMPOSIUM ON COMPUTERS AND COMMUNICATIONS (ISCC). *IEEE*. Cappadocia, Turkey, 2012. p. 637–642.

54 TOM'S HARDWARE. *Charts, benchmarks CPU Charts 2015, [04] SiSoftware Sandra 2015*. 2015. <http://www.tomshardware.com/charts/cpu-charts-2015/-04-SiSoftware-Sandra-2015,3696.html>. (Accessed on 05/02/2017).

55 TOM'S HARDWARE. *Charts, benchmarks CPU Charts 2015, [35] System Peak Power*. 2015. <http://www.tomshardware.com/charts/cpu-charts-2015/-35-System-Peak-Power,3727.html>. (Accessed on 05/02/2017).

56 DI, S.; KONDO, D.; CAPPELLO, F. Characterizing cloud applications on a google data center. In: 42ND INTERNATIONAL CONFERENCE ON PARALLEL PROCESSING. *IEEE*. Lyon, France, 2013. p. 468–473.

57 KARUNANITHI, A. K. et al. A survey, discussion and comparison of sorting algorithms. 2014.

APÊNDICE A – Resultados dos testes com *checkpointing* de tarefas

Neste apêndice são apresentados os gráficos de satisfação dos usuários, referentes aos resultados dos testes feitos com *checkpointing* de tarefas, excetuando-se o gráfico do caso S1M1CC que já foi apresentado anteriormente na figura 2. Conforme já foi apontado, os resultados aqui apresentados são muito semelhantes aos que foram mostrados anteriormente.

A.1 Resultados para o Modelo 1

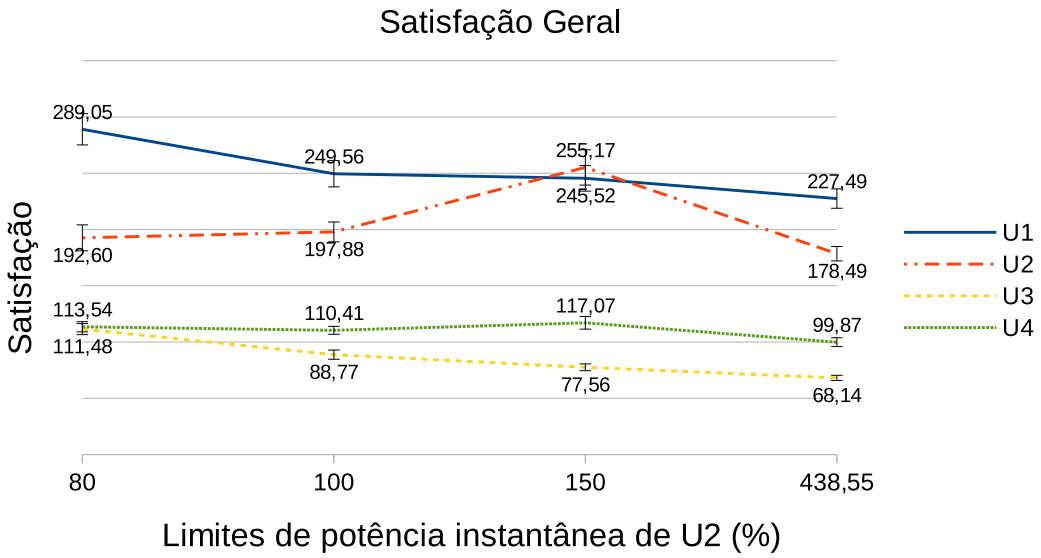

Figura 23 – Satisfação dos usuários (S2M1CC) com potência instantânea de U2 limitada e dos demais ilimitada

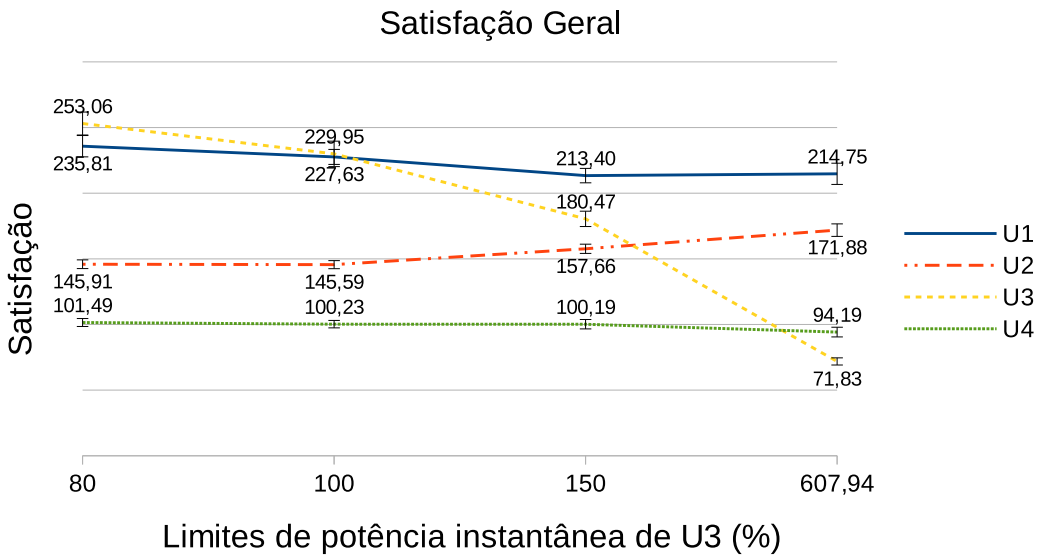

Figura 24 – Satisfação dos usuários (S3M1CC) com potência instantânea de U3 limitada e dos demais ilimitada

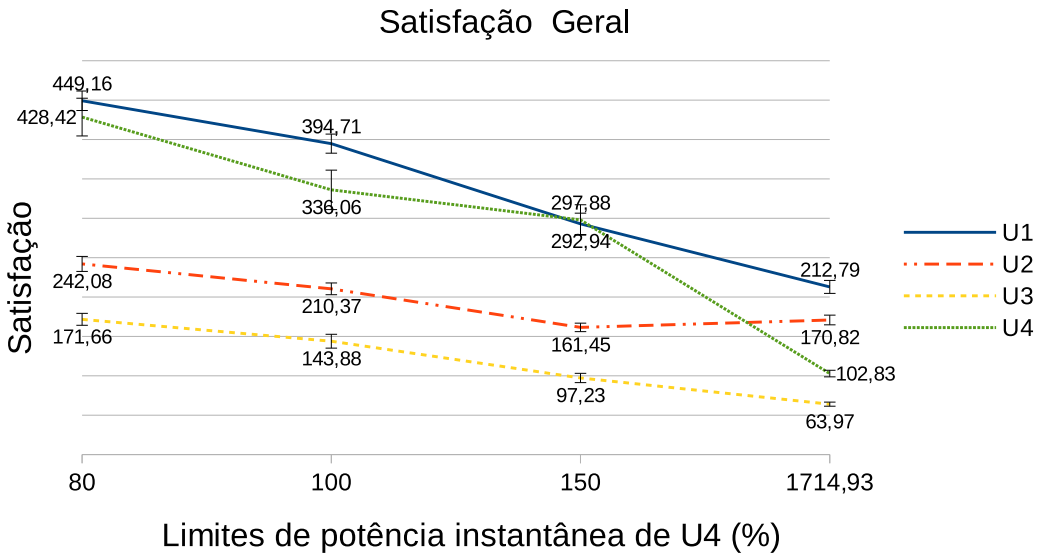

Figura 25 – Satisfação dos usuários (S4M1CC) com potência instantânea de U4 limitada e dos demais ilimitada

A.2 Resultados para o Modelo 2

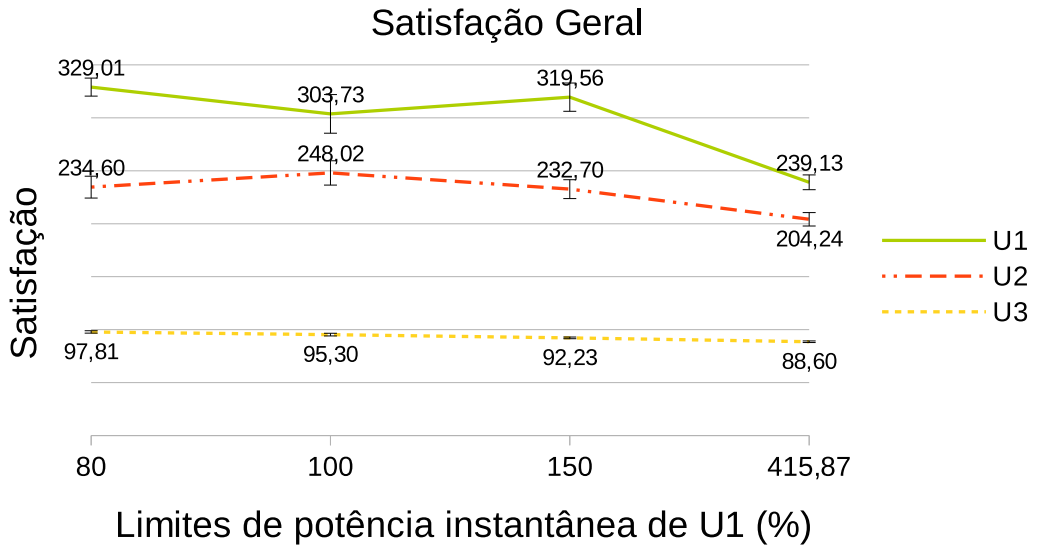

Figura 26 – Satisfação dos usuários (S1M2CC) com potência instantânea de U1 limitada e dos demais ilimitada

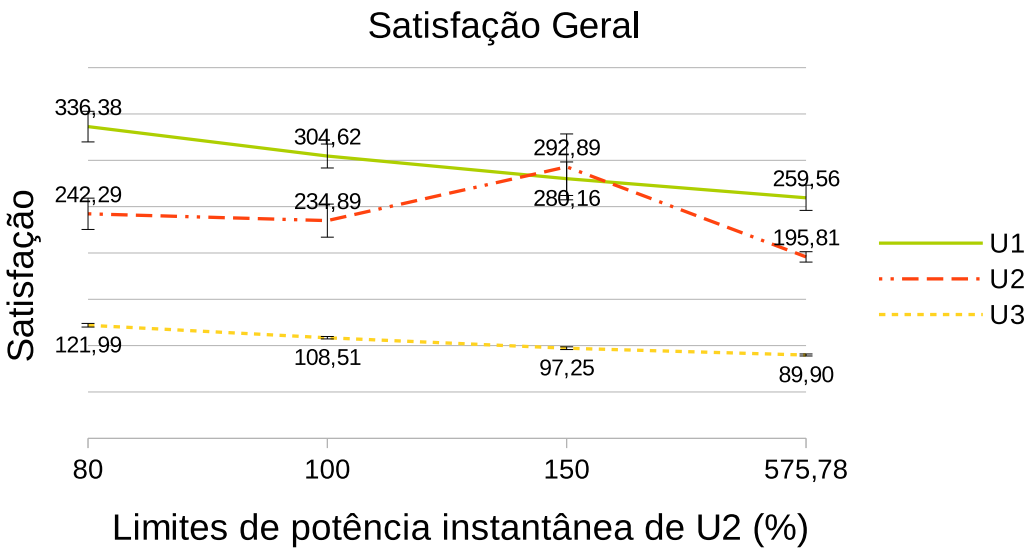

Figura 27 – Satisfação dos usuários (S2M2CC) com potência instantânea de U2 limitada e dos demais ilimitada

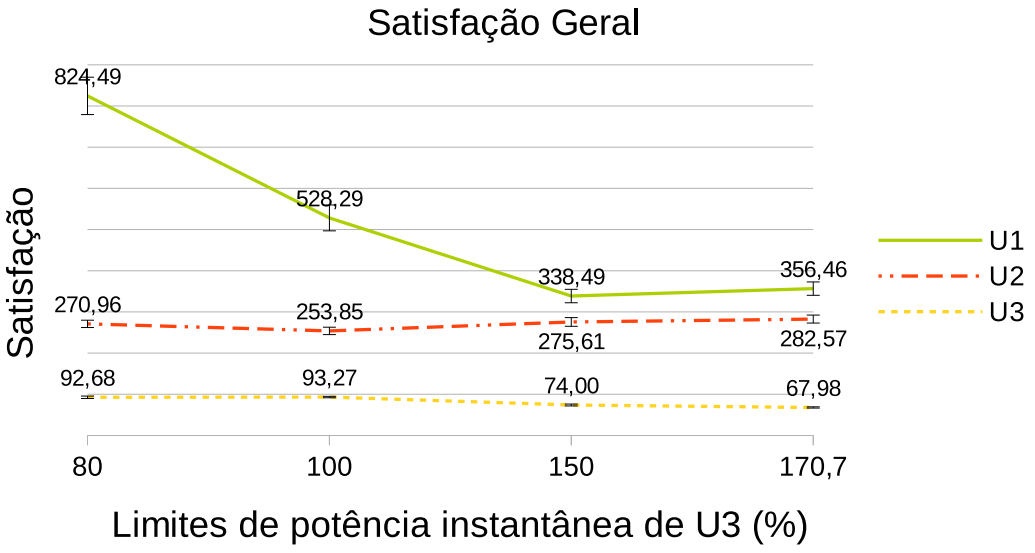

Figura 28 – Satisfação dos usuários (S3M2CC) com potência instantânea de U3 limitada e dos demais ilimitada