ESTRATÉGIAS BASEADAS EM POLÍTICAS VERDES PARA ALOCAÇÃO DE RECURSOS EM GRIDS COMPUTACIONAIS

Fábio José Coutinho da Silva

Tese de Doutorado apresentada ao Programa de Pós-graduação em Engenharia de Sistemas e Computação, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Doutor em Engenharia de Sistemas e Computação.

Orientador(es): Luís Alfredo Vidal de Carvalho

Rio de Janeiro

Abril de 2012

ESTRATÉGIAS BASEADAS EM POLÍTICAS VERDES PARA
ALOCAÇÃO DE RECURSOS EM GRIDS COMPUTACIONAIS

Fábio José Coutinho da Silva

TESE SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA (COPPE) DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE DOUTOR EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Examinada por:

_____

Prof. Luís Alfredo Vidal de Carvalho, D.Sc.

_____

Prof. Bruno Richard Schulze, D.Sc.

_____

Prof[a]. Roseli Suzi Wedemann, D.Sc.

_____

Prof. Felipe Maia Galvão França, Ph.D.

_____

Prof. Geraldo Bonorino Xexéo, D.Sc.

RIO DE JANEIRO, RJ - BRASIL
ABRIL DE 2012

Silva, Fábio José Coutinho da

Estratégias baseadas em políticas verdes para alocação de recursos em grids computacionais / Fábio José Coutinho da Silva. – Rio de Janeiro: UFRJ/COPPE, 2012.

XIII, 93 p.: il.; 29,7 cm.

Orientador: Luís Alfredo Vidal de Carvalho

Tese (doutorado) – UFRJ/ COPPE/ Programa de Engenharia de Sistemas e Computação, 2012.

Referências Bibliográficas: p. 71-79.

1. Alocação de Recursos em Grid. 2. Computação Verde. 3. Simulação de Ambientes de Grid. 4. Otimização Multiobjetivo. I. Carvalho, Luís Alfredo Vidal de. II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia de Sistemas e Computação. III. Estratégias baseadas em políticas verdes para alocação de recursos em grids computacionais.

Este trabalho é dedicado a meus pais, minhas irmãs,
meu filho e minha sobrinha.

AGRADECIMENTOS

Em primeiro lugar, gostaria de agradecer ao maestro deste Universo que concedeu a mim as oportunidades e condições para buscar essa importante conquista em minha vida.

Aos meus pais Getúlio e Nadja pelo sacrifício empenhado para garantir minha formação, por seus ensinamentos, suas palavras de conforto e esperança que não me deixaram fraquejar mesmo diante dos momentos mais difíceis dessa caminhada.

À minha irmã Andréa por sua preocupação, carinho, incentivo e apoio incondicional, que foram fundamentais para a finalização deste trabalho. Ainda não existem palavras para descrever a gratidão que tenho a ti.

À minha irmã Patrícia pelos bons momentos de descontração que passamos juntos e que muito contribuiu para aliviar o estresse do dia-a-dia.

Ao meu filho Fabinho e minha sobrinha Lara por serem os maiores responsáveis pela alegria em minha vida mesmo diante de muitas turbulências. Essas duas jóias raras formam o combustível necessário para vencer qualquer batalha.

Aos meus amigos Emmanuel, Thiago, Josué e Bebeca pelo apoio, carinho e incentivo que os colocam num lugar especial em meu coração.

À minha tia Nadjan por todo carinho dedicado a mim e por sempre estar ao meu lado em todas as etapas de minha vida, torcendo e rezando pelo meu sucesso.

À pessoa que talvez seja a maior torcedora dessa conquista. Karina, você realmente merece minha eterna gratidão. Caso fosse permitido somente escrever um único nome nesta página, esteja certa de que seria o teu. As tuas palavras e o teu apoio durante esta caminhada serão inesquecíveis.

À minha madrinha Ana Maria pelo carinho e por ter sido uma incentivadora de longa data.

Às minhas tias Norma, Naribel, Cristina, Uda, Fátima e meu avô João por torcerem pelo meu sucesso.

Aos amigos cariocas Arthur, Dayse e Marina (*in Memorian*) pelo apoio e carinho com que me receberam no Rio, sem a ajuda deles seria muito mais árdua esta caminhada.

Ao meu amigo Vinícius por sua lealdade, companheirismo e, principalmente, por ter se mantido sempre presente nos momentos de dificuldade e de alegria.

Ao meu amigo Leizer por seu companheirismo, dedicação e apoio acadêmico que muito contribuiu para este trabalho.

Ao meu amigo Jesus pelas informações privilegiadas e pela disposição em ajudar sempre.

Aos meus amigos Edgar e Marcelo pelos bons momentos de descontração.

Ao Renato Santana por seu envolvimento, apoio e discussões que contribuíram de forma significativa para o êxito deste trabalho.

Aos demais colegas do CBPF, prof. Javier Magnin, prof. Ignacio Bediaga, Hebert e Patrícia, pelo apoio e compreensão em diversos momentos em que estive ausente.

Agradeço imensamente ao meu orientador prof. Dr. Luís Alfredo por ter acreditado em mim e permitido que eu chegasse até aqui, apoiando-me em todas as etapas.

Aos professores Drs. Bruno Schulze, Roseli Wedemann, Geraldo Xexéo e Felipe França por se disporem a participar da banca de avaliação do doutorado.

Ao professor Claudio Bornstein pela valoroza contribuição e aos demais professores do PESC pelas excelentes aulas e também ao pessoal da secretaria pela atenção que sempre dispuseram a mim.

À Coppetec e à CAPES pelo suporte financeiro, bem como ao PESC e à CAT/CBPF que financiaram a apresentação de artigos resultantes desta tese.

Estendo ainda meus agradecimentos a todos aqueles que direta ou indiretamente foram incentivadores ou colaboradores deste trabalho.

Resumo da Tese apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Doutor em Ciências (D.Sc.)

ESTRATÉGIAS BASEADAS EM POLÍTICAS VERDES PARA ALOCAÇÃO DE RECURSOS EM GRIDS COMPUTACIONAIS

Fábio José Coutinho da Silva

Abril/2012

Orientador: Luís Alfredo Vidal de Carvalho

Programa: Engenharia de Sistemas e Computação

Nos últimos anos, a computação em grid tem se consolidado como uma solução capaz de integrar, em escala global, recursos geograficamente distribuídos e heterogêneos. Tal fato tem contribuído de maneira significativa para o aumento da infraestrutura de TI, haja vista o grid do projeto LHC, composto de centenas de sítios em mais de 30 países. Entretanto, toda essa malha computacional demanda um grande consumo de energia para se manter em funcionamento, o que tem gerado preocupações não apenas no aspecto financeiro, mas principalmente em relação ao impacto ambiental. Dados recentes mostram que a indústria de tecnologia da informação e comunicação tem sido responsável por aproximadamente 2% da emissão global de CO_2, equivalente a toda a indústria de aviação. Em virtude disso, a comunidade científica tem despertado grande interesse pela *computação verde*, a qual busca alternativas para atenuar prejuízos impostos ao meio ambiente pela área de TI, incluindo a redução do consumo de energia. Esta tese propõe estratégias para a alocação de recursos em grids, visando reduzir o consumo de energia em perspectiva global. As estratégias propostas foram avaliadas por meio de simulação e neste ambiente mostrou ser capaz de obter consideráveis níveis de economia de energia.

Abstract of Thesis presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Doctor of Science (D.Sc.)

STRATEGIES BASED ON GREEN POLICIES TO
RESOURCE ALLOCATION IN COMPUTING GRIDS

Fábio José Coutinho da Silva

April/ 2012

Advisor: Luís Alfredo Vidal de Carvalho

Department: Systems Engineering and Computer Science Program

In recent years, grid computing has consolidated itself as a solution able of integrating, on a global scale, geographically distributed and heterogeneous resources. This fact has significantly contributed to the increase in infrastructure of IT, as seen on the grid of the LHC project, composed of hundreds of sites in over 30 countries. However, all this computer power requires a large energy consumption to keep in working; such fact has raised concerns not only in economics issue, but mainly in considering to the environmental impact. Current data show that the information technology and communication industry has been in charge of about 2% of carbon dioxide global emission, equivalent to the entire aviation industry. As a result, research community has greatly interested in green computing, which seeks alternatives to attenuate environmental damages imposed by the IT area, including the energy consumption reduction. This work proposes strategies for resource allocation in grids to reduce energy consumption in a global perspective. The proposed strategies has been evaluated by simulation and in this environment proved to be able to obtain significant levels of energy savings.

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1 – Introdução

Nos últimos anos, o mundo tem se voltado para questões relacionadas à preservação do meio ambiente. Organizações não governamentais, instituições de pesquisa, grandes empresas, políticos e demais setores da sociedade têm buscado soluções que permitam continuar a escalada de desenvolvimento atual sem comprometer o meio ambiente e as necessidades das gerações futuras, ou seja, *o crescimento sustentável*.

Dentre as questões envolvidas na discussão ambiental, a redução da emissão de gases poluentes à atmosfera, tal como o CO_2, ocupa um lugar de destaque. Atualmente, sabe-se que a matriz energética mundial contribui significativamente para o efeito estufa, uma vez que boa parte de sua produção ainda depende da queima de combustíveis fósseis (CARDOSO, 2006).

Na área da computação, a preocupação com a redução do consumo de energia esteve, inicialmente, restrita aos dispositivos móveis, *laptops* e sistemas embarcados, buscando sempre aumentar o tempo de carga das baterias. Entretanto, mais recentemente, esta visão tem sido ampliada. E tal fato deve-se, em muito, à evolução da computação de alto desempenho, o que levou a um consequente aumento da demanda por consumo de energia.

Um exemplo que caracteriza bem essa evolução vem da computação em grid, que nas últimas décadas, tem se consolidado como uma solução capaz de integrar, em escala global, recursos geograficamente distribuídos e heterogêneos. De fato, muitos projetos de e-Ciência têm adotado grids como plataforma para execução de seus experimentos. LHC (WLCG, 2002) e LIGO (LIGO, 2000) são exemplos bem sucedidos de experimentos na área da Física que utilizam a computação em grid. Além da física, grids computacionais são utilizados em diversas outras áreas, tais como bioinformática, geologia, biomedicina, climatologia, astronomia, entre outros.

Apesar da indústria de hardware está caminhando na direção de priorizar a eficiência energética, somente isso, ainda não é suficiente a curto prazo. Assim sendo, nos dias de hoje, a gerência eficiente do consumo de energia tem se tornado uma busca constante no âmbito da computação distribuída. Grids de escala global são prioritários para minimizar o consumo de energia. O grid do projeto LHC, por exemplo, consome aproximadamente 2,5 MW apenas para manter o centro computacional de seu sítio no

CERN (considerado *Tier-0*) (CERN, [S.d.]), sem levar em conta a energia gasta com refrigeração.

Diante deste cenário, os pesquisadores da área de informática são chamados a desenvolver soluções computacionais que busquem minimizar o consumo da energia gasta para manter em operação ambientes de computação de alto desempenho, tal como grids.

Em resposta a esse apelo, este trabalho propõe algumas estratégias para a alocação de recursos em ambientes de grid com o intuito de minimizar o consumo global de energia. A alocação de recursos é um problema clássico da computação em grid, o qual visa escolher que recursos alocar para a execução de uma determinada tarefa ou *job*.

Tradicionalmente, o processo de alocação de recursos em grid leva em conta diferentes aspectos, tais como: CPU, memória, disco, largura de banda, histórico de execuções anteriores, entre outros. Neste trabalho, foi considerado um requisito adicional: *a eficiência energética*. Dessa maneira, a execução das tarefas também passou a ser orientada por políticas de computação verde.

Portanto, resumidamente, este trabalho tem como objetivo geral *propor e avaliar estratégias para a alocação de recursos em grid a fim de promover a redução de energia por meio de políticas de computação verde*. Esse propósito é decomposto em tópicos mais específicos, os quais serão explicados no decorrer da apresentação do trabalho. Sendo assim, este documento encontra-se organizado da maneira descrita a seguir.

O capítulo 2 discute os principais conceitos acerca dos temas que servem de fundamentação paro o entendimento desta tese.

O capítulo 3 traz um estudo a respeito dos sistemas de gerência em grids e discute trabalhos da computação de alto desempenho que tratam do escalonamento sob o enfoque da redução de energia.

O capítulo 4 trata dos conceitos envolvidos na definição de um modelo para a representação do consumo de energia em recursos do grid.

O capítulo 5 discute aspectos relacionados à arquitetura do grid considerado neste trabalho e apresenta conclusões acerca do ambiente, que podem interferir nas estratégias de alocação.

O capítulo 6 explica em detalhes as estratégias propostas para a alocação de recursos, além de apresentar suas respectivas soluções algorítmicas. A organização do capítulo prevê cada estratégia contida numa seção principal.

O capítulo 7 discute os aspectos envolvidos na avaliação das estratégias propostas. Além disso, define cenários de execução e analisa os resultados encontrados pela aplicação das estratégias frente a tais cenários.

Finalmente, no capítulo 8 são apresentadas as conclusões deste trabalho, contendo suas contribuições, dificuldades e sugestões para trabalhos futuros.

Capítulo 2 – Fundamentação

Este capítulo analisa os principais conceitos relacionados ao tema abordado nesta tese. Incialmente, discute-se o processo de alocação de recursos em grids, em seguida, são apresentados os conceitos da computação verde (originado do termo, em inglês: *green computing*) e sua influência sobre a computação de alto desempenho (HPC).

2.1 Computação em Grid

A computação em grid ou computação em grade (do termo em inglês: *grid computing*) surgiu em meados dos anos noventa. FOSTER e KESSELMAN (1998) enunciaram uma das primeiras definições sobre grid computacional, na qual o conceituaram como uma *infraestrutura de hardware e software capaz de fornecer acesso confiável, ubíquo, consistente e barato a funcionalidades computacionais de alto nível*. Posteriormente, a definição foi refinada, atribuindo-se ao grid a função de compartilhamento de recursos coordenados para solucionar problemas em organizações virtuais (VOs) multi-institucionais e dinâmicas (FOSTER *et al.*, 2001). Esta última definição enxerga o grid com um nível de compartilhamento que permite acesso direto a computadores, *software*, dados, e demais recursos controlados por organizações virtuais. Tal visão é compatível com a organização de grandes projetos que envolvem a colaboração entre instituições de pesquisa para a execução de experimentos científicos.

Mais recentemente, BUYYA e VENUGOPAL (2005) definiram grid como um tipo de sistema distribuído e paralelo que permite o compartilhamento, seleção e agregação de recursos autônomos dinamicamente, considerando características tais como: disponibilidade, capacidade, desempenho, custo e requisitos de QoS (*Quality of Service*) dos usuários.

Desta forma, pode-se dizer que a tecnologia de grid tem caminhado na direção de atender a três principais objetivos:

- Obter desempenho maior do que em supercomputadores ou clusters, através de uma infraestrutura concebida para prover paralelismo;
- Ser capaz de explorar recursos ociosos (ciclos de instrução) de vários computadores conectados através de redes WANs;

- Permitir que diversos usuários distribuídos geograficamente possam trabalhar remotamente de forma colaborativa.

De acordo com sua organização e finalidade, grids computacionais podem ser classificados em dois principais tipos: grid de estação de trabalho (*desktop grid)* e *grid* computacional. Esses dois tipos de grid possuem infraestrutura de software e modelos de organização distintos.

Em grids de estação de trabalho, as aplicações costumam ser divididas em atividades menores que executam a partir de recursos ociosos, seguindo o modelo também conhecido como computação voluntária. A computação voluntária ocorre quando proprietários de computadores (normalmente, cidadãos comuns) doam recursos computacionais (processamento ou armazenamento) para um ou mais projetos. A ideia consiste em permitir a utilização desses recursos pelo projeto enquanto os computadores estiverem ociosos, ou seja, no tempo em que seus proprietários não o estejam usando. O projeto SETI@home (SETI@HOME, [S.d.]) foi um dos primeiros grids construídos sobre esse modelo, tendo como finalidade processar dados obtidos a partir de telescópios espaciais.

Diferentemente dos grids de estação de trabalho que obtêm recursos de pessoas comuns, grids computacionais possuem infraestrutura de larga escala, tais como *clusters* e supercomputadores. O modelo de organização também é oposto, em grids computacionais têm-se mais usuários que doadores de recursos. Além disso, grids computacionais fazem uso de uma infraestrutura de software mais complexa, através do uso de mediadores de grid, tais como: gLite (GLITE, [S.d.]) e Globus (GLOBUS, [S.d.]).

Neste trabalho, são analisadas soluções de escalonamento para grids do tipo computacional que atuam em escala global. Sendo assim, a computação voluntária e os grids de estação de trabalho estão fora do escopo abordado por esta linha de trabalho.

2.2 **Alocação de Recursos em Grids**

Em termos gerais, o problema do escalonamento de tarefas em ambientes distribuídos configura-se como um problema NP-completo (ULLMAN, 1975), onde soluções baseadas em busca exaustiva são inviáveis devido ao alto custo computacional requerido.

O problema do escalonamento vem sendo estudado mesmo antes do surgimento da tecnologia de grid (KWOK, AHMAD, 1999), resultando grande parte em soluções baseadas em heurísticas. De fato, alocar recursos computacionais para a execução de tarefas tem sido um problema analisado em diferentes plataformas (P2P, HPC, etc.). Contudo, ao considerar o ambiente de grid, o escalonador necessita avaliar novas variáveis que retratam o comportamento dinâmico e heterogêneo do grid. Alguns desafios lançados pela arquitetura do grid computacional são (YU *et al.*, 2008):

- Muitos usuários competindo por recursos;
- Os recursos do grid não estão sob o controle do escalonador;
- Os recursos são heterogêneos e podem não executar identicamente a mesma tarefa;
- Muitas aplicações requerem a movimentação de grandes volumes de dados entre múltiplos sites do grid.

De acordo com YU *et al.* (2008), os escalonadores de workflow em grids podem ser classificados em dois grupos principais de acordo com o tipo de escalonamento: escalonamento baseado em melhor esforço e escalonamento baseado em restrições de QoS. O primeiro tem como objetivo minimizar o tempo final de execução do workflow, sendo aplicável a grids comunitários (baseados em VOs). Enquanto o último é voltado para grids orientados a serviços (grids utilitários) e visa garantir as restrições de QoS dos usuários.

Seguindo a classificação citada anteriormente, pode-se afirmar que o escalonador proposto neste trabalho encontra-se alinhado ao escalonamento baseado em melhor esforço. Contudo, é importante ressaltar que os escalonadores tradicionais costumam utilizar estratégias dirigidas a minimizar tempo (*makespan*) e/ou custo de execução do workflow, ao passo que a proposta deste trabalho lança olhar sobre um novo objetivo: *minimizar a energia consumida pelos recursos do grid para executar as tarefas com o mínimo de degradação para o grid e o meio ambiente*.

A seção 3.1 discute as principais propostas de escalonamento de tarefas fornecidas pelos sistemas de gerência de workflows em grids conhecidos.

2.3 **Computação Verde**

Estima-se que em 2006, o consumo de energia elétrica por servidores e CPDs (Centro de Processamento de Dados) nos Estados Unidos atingiu a marca de 61 bilhões de quilowatt-hora (kWh). Isso equivale a, aproximadamente, 1,5% do total de energia consumida pelo país naquele ano, e representa um custo de 4,5 bilhões de dólares (BROWN *et al.*, 2007). Outro aspecto relevante desse estudo é o fato de o consumo de energia com equipamentos de TI em 2006 ter sido superior ao dobro do consumido em 2000. O relatório também indica uma estimativa de consumo para o ano de 2011 que ultrapassa a marca de 100 bilhões kWh.

O consumo crescente apontado por esse estudo reflete a magnitude da energia elétrica consumida em CPDs. Tal realidade preocupa não apenas pela questão financeira, mas principalmente pelo consequente aumento da emissão de dióxido de carbono (CO2). Diversos estudos têm evidenciado que a utilização de energia colabora para o aumento da emissão de CO2, seja diretamente em sua produção ou mesmo de forma indireta. É importante destacar que até mesmo a energia produzida por hidrelétricas, tida no passado como uma energia limpa, atualmente, sabe-se que produz quantidade considerável de CO2 (ROSA *et al.*, 2002). De acordo com recente estimativa da Gartner (GARTNER, [S.d.]), a indústria da tecnologia da informação e comunicação é responsável por aproximadamente 2% da emissão global de dióxido de carbono (GARTNER NEWSROOM, 2007). Tal porcentagem já equivale à emissão de CO2 de toda a indústria de aviação.

Desta forma, o uso eficiente de energia pela área de TI converge para um dos mais importantes desafios encarados pela humanidade neste século: o desenvolvimento de tecnologias que permitam o crescimento de forma sustentável. Diante deste cenário, promover a computação verde tem sido um tema bastante discutido tanto pela comunidade acadêmica quanto pela corporativa (MINGAY, 2007).

A computação verde tem se consolidado como o estudo e a prática de projetar, fabricar, usar e descartar computadores, servidores e subsistemas associados de forma eficiente e causando o mínimo ou nenhum impacto ao meio ambiente (MURUGESAN, 2008).

Atualmente, computação verde representa um tema de cobertura bastante ampla com muitas posições ainda em aberto e que pode ser explorado a partir de diferentes perspectivas. A seguir, são apresentadas algumas aplicabilidades de TI verde:

- *eficiência algorítmica* – busca-se projetar algoritmos mais eficientes com o intuito de não apenas reduzir o tempo de execução como também a demanda por recursos computacionais.
- *virtualização* – apresenta-se como uma prática interessante para a implementação em CPDs por permitir que diversos sistemas físicos (*hardaware*) sejam substituídos por máquinas virtuais, as quais executarão em um único equipamento, resultando em economia de energia considerável (BARHAM *et al.*, 2003).
- *servidor de terminais* – tem por objetivo reduzir o montante de energia consumido por uma típica estação de trabalho cliente, empregando o conceito de "cliente magro" (*thin-client*).
- *gerência de consumo de energia via software* – implementa soluções para a redução do consumo de energia em equipamentos de TI (por exemplo: monitores e discos) através de controle por software durante períodos de inatividade.
- *mídias de armazenamento* – corresponde à estratégia de substituir as tradicionais mídias de armazenamento (HDs e fitas) por dispositivos SSD (unidades de estado sólido) tal como a *memória flash*. SSDs não possuem partes móveis, por conta disso, o consumo de energia e a necessidade de resfriamento são bastante inferiores aos de discos comuns (MEARIAN, 2009).
- *processadores* – busca-se fabricar processadores com maiores taxas de desempenho por watt (FLOPS/watts) e, consequentemente, mais econômicos no consumo energético (AMD, [S.d.])(INTEL, [S.d.]).
- *descarte inteligente* – tem como finalidade encontrar a maneira adequada para se desfazer dos equipamentos. Algumas práticas, como reciclagem e doação, são almejadas em vez de simplesmente jogar em aterros sanitários comuns, onde pode haver risco de contaminação do solo e da água em consequências das substâncias químicas contidas nos hardwares.

Um recente artigo da revista *Smashing* (BINNEY, 2010) comenta o trabalho do físico Alexander Wissner-Gross, professor da Universidade de Harvard. O professor conclui que o ato de navegar em websites mais pesados (contendo animações e vídeos)

pode gerar cerca de 300mg de CO2 por segundo. Para chegar a esse número, o pesquisador contabiliza a eletricidade consumida pelo computador do visitante, seguido pela infraestrutura de rede para transmissão dos dados e uma parcela menor pelos servidores e data centers que abrigam o site. Outro trabalho relevante desenvolvido pelo físico é o projeto CO2Stats, o qual busca ensinar boas práticas de computação verde a desenvolvedores de sítios e *blogueiros*. CO2Stats emite certificados para websites que estejam em conformidade com políticas de computação verde (WISSNER-GROSS, [S.d.]).

O esforço desse pesquisador representa mais um passo na direção de identificar e privilegiar os colaboradores da computação verde, além de compartilhar a responsabilidade com o usuário final, o internauta. Afinal, o mesmo também é parte envolvida no processo. Segundo MURUGESAN (2008), cada PC em uso gera em média aproximadamente uma tonelada de CO2 por ano.

Portanto, fica evidente que a busca por um ambiente sustentável é algo que só pode ser obtido através da colaboração e responsabilidade entre todas as partes envolvidas. Em se tratando de ambientes HPC, a responsabilidade torna-se ainda maior haja vista a elevada demanda energética para suprir seu funcionamento.

2.4 **Computação Verde aplicada a HPC**

Conforme discutido na seção anterior, uma grande quantidade de energia vem sendo consumida em instalações de TI espalhadas pelo mundo. Os CPDs têm se tornado mais poderosos a cada dia, dispondo de *clusters* ou supercomputadores capazes de manter e processar grandes volumes de dados. Tal fato contribui para o aumento do consumo de energia elétrica, o qual se torna significativo principalmente em duas vertentes: a refrigeração do ambiente e o funcionamento dos equipamentos computacionais. O consumo de energia com a iluminação dos CPDs tem sido considerado irrelevante para as pesquisas interessadas em computação verde. No âmbito da solução desenvolvida no decorrer deste trabalho, considera-se apenas a redução da energia consumida pelos equipamentos de computação, desconsiderando a energia dispendida com a refrigeração do ambiente.

Um exemplo da importância da computação verde em HPC tem sido a criação da lista Green500 (GREEN500, [S.d.]) (SHARMA *et al.*, 2006), a qual anuncia, a cada seis meses, os supercomputadores com maior eficiência energética (MFLOPS/Watt). A

busca por reconhecer e privilegiar a utilização de recursos comprometidos com a computação verde tem sido constante em ambientes HPC, tendo sido alvo de órgãos governamentais, instituições de pesquisa e grandes corporações.

Atualmente, a redução do consumo de energia em grids globais tem sido encarada como um desafio de alta prioridade (MEINHARD, 2008). Sob o ponto de vista da computação verde, cada uma das máquinas que fazem parte do grid representa um possível alvo para a aplicação de práticas que visem reduzir o consumo de energia. De fato, a presença de recursos computacionais heterogêneos, característica marcante de grids, permite que recursos com diferentes níveis de consumo de energia operem ao mesmo tempo.

Todavia, em um grid de escala global torna-se impossível garantir que todos os sítios disponibilizem apenas recursos comprometidos com políticas de computação verde. Normalmente, em grids globais, as instituições pertencentes ao grid são responsáveis por manter de forma autônoma seus próprios recursos de TI. Apesar disso, é possível analisar a eficiência energética dos recursos do grid, através da publicação de informações referentes ao consumo de energia das máquinas de cada sítio participante do grid.

Em se tratando de grids utilitários, um caminho fértil para a exploração de computação verde é através do uso da tecnologia de virtualização, a qual é inerente ao modelo desse tipo de arquitetura. A principal desvantagem obtida pelo uso da virtualização é a sobrecarga (*overhead*) gerada pelas migrações das máquinas virtuais.

Outra técnica bastante explorada para o gerenciamento de energia em ambientes HPC é a DVS (*Dynamic Voltage Scaling*) (SEMERARO *et al.*, 2002). A aplicação de DVS em recursos computacionais permite que se diminua o desempenho da CPU através da redução explícita e intencional da voltagem e frequência da CPU (processo também chamado de *undervolting*), consequentemente, resultando na queda do consumo de energia pelo recurso. DVS encontra-se disponível para as CPUs mais modernas e tem sido frequentemente aplicada em dispositivos móveis, laptops e servidores.

Atualmente, diversos trabalhos utilizam o conceito de DVS em suas propostas para aplicação de computação verde em HPC. Entretanto, a aplicação desta técnica em grids globais através de um controle externo (fora de cada sítio) vai de encontro à autonomia exercida pelos sítios que compreendem o grid. No modelo de grid considerado neste trabalho, os sítios participantes colocam seus recursos à disposição

para a execução das aplicações do grid. Contudo, a gerência e manutenção dos recursos ficam a cargo de cada sítio.

Deste modo, torna-se inviável prover uma solução global baseada em DVS para reduzir o consumo de energia do grid como um todo sem romper um princípio do funcionamento do grid-— a autonomia dos sítios.

Capítulo 3 – Trabalhos Relacionados

Neste capítulo são discutidos relevantes trabalhos encontrados na literatura relacionados com a proposta descrita nesta tese. Inicialmente, a seção 3.1 apresenta uma análise sobre os principais Sistemas de Gerência de Workflows Científicos (SGWfCs). Em seguida, a seção 3.2 discute as soluções de mediadores de grid para o problema do escalonamento e uma análise frente à computação verde.

O levantamento feito a partir da análise desses trabalhos visou não apenas conhecer as principais técnicas de escalonamento, mas também permitiu conhecer o estado-da-arte acerca da computação verde aplicada a HPC.

3.1 **Escalonamento em SGWfCs**

A alocação de recursos computacionais em ambiente de grid tem sido um tópico de intensa pesquisa pela comunidade científica. Uma técnica pioneira empregada na alocação de recursos é o "casamento" (*matching*), onde provedores publicam as características de seus recursos, as quais são casadas com requisições dos usuários (THAIN *et al.*, 2001).

Inicialmente, os trabalhos concentravam-se apenas na maximização do *throughput* baseando-se em modelos parametrizados por informações dos recursos (CPU, memória, disco e largura de banda) e propriedades quantitativas do comportamento do job (ex. tempo de espera na fila e tempo de execução) (DOWNEY, 1997) (SMITH *et al.*, 1999). Mais recentemente, alguns trabalhos ainda seguem esta direção (SONMEZ *et al.*, 2009) (LINGRAND *et al.*, 2008) enquanto outros atuam na especificação de modelos econômicos e inteligentes (aprendizagem de máquina) de provisionamento de recursos por qualidade de serviço (Quality of Service - QoS) (LINGRAND *et al.*, 2008) (PEREZ *et al.*, 2008).

Nas seções subsequentes são discutidas as abordagens dos principais SGWfCs, enfatizando suas estratégias para mapeamento de workflows em recursos do grid e analisando o possível emprego de políticas de computação verde.

3.1.1 Taverna

Taverna (OINN *et al.*, 2006) é um SGWfC de código aberto desenvolvido em linguagem Java pelo Projeto myGrid (MYGRID, [S.d.]). Possui enfoque voltado para a bioinformática, apesar de ser também utilizado em outros domínios (ex. astronomia). O

projeto teve início em 2001, através de um consórcio que reunia universidades e institutos do Reino Unido, sendo um dos pioneiros no desenvolvimento da tecnologia de workflow científico.

A proposta inicial do myGrid previa o suporte à execução de experimentos *in-silico* em ambientes de grid. Todavia, suas principais contribuições não abordam com profundidade questões sobre gerência e execução em grids. Em vez disso, Taverna tem dedicado esforço para oferecer ao pesquisador melhorias na fase de construção (*design*) do workflow, além de oferecer serviços de alto nível que exploram proveniência de dados (STEVENS *et al.*, 2007), descoberta de recursos/serviços/workflows (WROE *et al.*, 2007), gerência de metadados (BELHAJJAME *et al.*, 2008b) e ontologias (PREECE *et al.*, 2008). Taverna, ainda inclui uma ferramenta que utiliza anotações semânticas associadas aos serviços e faz uso de técnicas de inferência para as anotações de serviços (BELHAJJAME *et al.*, 2008a).

Por meio do projeto BioCatlogue (BHAGAT *et al.*, 2010) foram incorporadas as experiências dos componentes Taverna Registry e myExperiment para a construção e gerência de um catálogo de serviços web para *Life Sciences*, disponibilizando atualmente cerca de 2000 serviços. Os serviços do catálogo possuem conteúdo descritivo e podem ser buscados através de múltiplas maneiras baseadas em categorias, *user tags*, provedores de serviço e dados de entrada/saída.

O escalonamento de workflows no ambiente Taverna ocorre sem que haja a análise dos recursos do grid orientada por alguma estratégia para tomada de decisão (por exemplo, a análise de características dos recursos). Em vez disso, Taverna deixa a cargo do usuário a seleção de recursos ou serviços em que ocorrerá a execução das tarefas do workflow. Ainda é facultada ao usuário a possibilidade de alocar um conjunto de recursos para a execução de uma tarefa. Dessa maneira, em caso de falha em um recurso, a tarefa passa a ser executada em um recurso alternativo automaticamente invocado.

3.1.2 Kepler

Kepler (LUDÄSCHER *et al.*, 2006) é um SGWfC de código aberto que estende o sistema Ptolemy II, desenvolvido em linguagem Java pela Universidade de Berkeley. A fase de construção (design) de workflows em Kepler faz uso do paradigma da modelagem orientada a atores (herdada do sistema Ptolemy), onde um workflow é composto de um *diretor* e um conjunto de *atores*. O diretor representa um componente

que permite ao projetista do workflow estabelecer controle sobre a execução do mesmo. Enquanto os atores representam as tarefas que compõem o workflow. Ou seja, pode-se dizer que o diretor define a estrutura do workflow, através da orquestração dos atores presentes.

Kepler dispõe de vários tipos de diretores em seu ambiente. Através do diretor PN (*Process Network*), os atores são executados como processos independentes que executam concorrentemente. Já o diretor SDF (*Synchronous Data-Flow*) permite um único ator executando por vez, de forma sincronizada, garantindo a ausência de *deadlocks*. O diretor DDF (*Dynamic Dataflow*) é usado para workflows que requerem *loops* ou estruturas de controle. Ainda existem outros diretores com propostas mais específicas, tais como DE (*Discrete Event systems*) e CT (*Continuous-Time models*). Os diretores PN, SDF e DDF são similares aos conceitos de paralelismo, sequência e *choice/iteration* apresentados na conhecida taxonomia de SGWfC para grids (YU, BUYYA, 2005), respectivamente.

No que diz respeito a mapeamento de recursos, similarmente ao Taverna, Kepler permite que o pesquisador selecione os recursos que serão usados para a execução. Cabe à máquina de execução prover um escalonamento para transferência de dados e execução dos jobs sobre os recursos escolhidos. Este escalonamento baseia-se nos modelos de execução e abstrato definidos previamente. Kepler também pode ser usado para configurar e submeter jobs para uma variedade de outros sistemas de grid, tais como Griddles (KOMMINENI, ABRAMSON, 2005) e Nimrod (ABRAMSON *et al.*, 1995).

3.1.3 Vistrails

Vistrails (BAVOIL *et al.*, 2005) é um SGWfC que tem como diferencial o suporte à visualização gráfica dos resultados dos workflows. Trata-se de um sistema de código aberto desenvolvido em linguagem python pela Universidade de Utah. Em Vistrails, usuários podem criar e editar workflows através de uma interface gráfica denominada *Vistrails Builder*. Para cada workflow é gerada uma especificação representada em XML, o que facilita o reúso e simplifica a validação dos workflows através de *parsers*.

O workflow criado em Vistrails é formado por um conjunto de módulos que representam as tarefas do workflow. Cada módulo pode conter uma ou mais funções. Vistrails emprega a noção de um workflow evolutivo, mantendo as diversas versões de

um workflow. Através do histórico, é facultada ao cientista a possibilidade de comparar os resultados obtidos entre duas versões do workflow executado. Para ajudar na comparação é disponibilizado o componente *Visualization Spreadsheet*, o qual fornece uma interface para visualização dos resultados.

Vistrails possui ainda um componente denominado Gerente de Cache, responsável por atender às requisições para execuções de workflows. A sua principal função é fazer uma análise do workflow em busca de módulos que tenham sido previamente executados, considerando os parâmetros submetidos. Um atributo denominado *cacheable* indica se deve ser mantido ou não um cache para os resultados de um módulo. Caso tais resultados sejam encontrados, são recuperados em cache e compartilhados com todos os workflows, evitando tempo desperdiçado com possíveis re-execuções.

Apesar de ter produzido relevantes contribuições para a visualização e proveniência de dados, Vistrails não se apresenta habilitado para a execução sobre recursos geograficamente distribuídos, como em grids. Não tendo sido encontrado até o presente momento solução de escalonamento ou estratégia para mapeamento de recursos. Os autores argumentam a condição de *grid-enabled* através do suporte e uso de serviços web.

3.1.4 Pegasus

Pegasus (DEELMAN *et al.*, 2004a) (DEELMAN *et al.*, 2004b) é um sistema desenvolvido pelo projeto GriPhyN (Grid Physics Network) cujo objetivo principal é mapear workflows abstratos em recursos do grid. Pegasus utiliza a *engine* DAGMan (FREY *et al.*, 2001) para controlar a execução dos workflows.

A entrada para o sistema Pegasus consiste no formato DAX (DAG XML). DAX é dividido em três partes: *lista de arquivos* – contém todos os arquivos usados no workflow, os quais podem ser do tipo entrada, saída, entrada/saída e executável; *tarefas* – contém todas as tarefas que compreendem o workflow a ser executado; *dependências* – contém as dependências existentes entre as tarefas do workflow.

Pegasus carece fortemente de suporte às etapas de construção, edição e validação de workflows. Dessa forma, o mesmo prevê a possibilidade de que seus workflows sejam definidos através do sistema Chimera (FOSTER *et al.*, 2002), o qual combina esquema e linguagem (VDL – *Virtual Data Language*) para prover a representação, consulta e derivação de dados automaticamente. Ainda assim, Chimera e

Pegasus carecem de ferramentas gráficas para auxiliar o cientista na construção dos workflows. O sistema Chimera também requer do usuário um certo conhecimento sobre a linguagem VDL. Uma solução para suprir essa carência, é viabilizar a integração de Pegasus com outros SGWfCs, tal como a solução proposta para integrar Kepler e Pegasus (MANDAL *et al.*, 2007).

Pegasus utiliza os serviços MDS e RLS do Globus para encontrar os recursos do grid e descobrir quais réplicas estão disponíveis, respectivamente. A partir do processamento das informações obtidas junto a esses serviços, Pegasus desenvolve algumas estratégias de mapeamento de workflows que aplicam o particionamento em diferentes granularidades: mapeamento de workflows inteiros; mapeamento de porções do workflow e mapeamento de tarefas individuais do workflow. Na visão mais simples, as tarefas são mapeadas individualmente, e o ambiente Pegasus provê interface para um escalonador definido pelo usuário, disponibilizando alguns algoritmos de escalonamento tais como Min-Min e GRASP (BLYTHE *et al.*, 2005).

Considerando o particionamento do workflow definido pelo cientista em sub-workflows, tem-se que cada sub-workflow será mapeado pelo Pegasus, sendo a ordem do mapeamento determinada pelas dependências entre os sub-workflows. Em alguns casos, os sub-workflows podem ser mapeados e executados em paralelo.

Após o mapeamento, Pegasus pressupõe o uso da máquina de execução DAGMan (FREY *et al.*, 2001) para gerenciar o workflow executável (com recursos identificados) resultante do processo de mapeamento. Assim, DAGMan utiliza Condor ou Condor-G para submeter tarefas do workflow a partir de um host para uma máquina local, um cluster, ou um grid.

3.1.5 Triana

Triana (TRIANA, [S.d.]) (TAYLOR *et al.*, 2003) consiste de um ambiente de desenvolvimento de workflows desenvolvido pela Universidade de Cardiff. Este SGWfC faz uso do Grid Application Prototype (GAP), uma API genérica que provê um subconjunto das funcionalidades da interface Grid Application Toolkit (GAT), implementada pelo projeto GridLab (ALLEN *et al.*, 2002). A proposta do Triana prevê o uso do GAP como uma plataforma independente de mediador para o desenvolvimento de aplicações de grid através da implementação de três *bindings*: JXTA, web services (WSRF) e P2Ps.

Os workflows modelados no Triana são compostos de unidades que correspondem a aplicações. O cientista tem a opção de utilizar unidades já disponibilizadas na interface ou pode construir suas próprias unidades, ou seja, incluir suas próprias aplicações. O workflow pode então ser montado pelo usuário através do uso de interface *drag-and-drop* num editor visual.

Triana possui uma arquitetura modularizada e plugável, sendo composta por um conjunto de componentes flexíveis que atuam em cooperação. O componente *Triana GUI* é um cliente (*light weight*) que se conecta ao componente *Triana Engine* (TE), localmente ou através da rede. Dessa maneira, usuários podem utilizar o componente *Triana Controlling Service* (TCS) para construir e executar seus workflows, além de visualizar seus resultados. Cada TCS interage com um ou mais *Triana Service* (TS), sendo este último capaz de executar grafos de tarefas parciais ou completos, localmente ou através de distribuição.

A política de distribuição de tarefas baseia-se no conceito de unidades *Triana Group* (TG), onde TG representa um agrupamento de unidades (ferramentas) que podem ser distribuídas ao longo do grid. Triana implementa duas políticas de distribuição para um TG: paralela e *pipeline*. A primeira distribui as unidades da TG entre nós distintos, de forma independente, sem haver comunicação entre os nós. Já na política de *pipeline* os dados intermediários são passados entre os nós e a distribuição das unidades de um TG segue uma visão hierárquica.

O mapeamento de recursos em Triana passa pelo uso do GAT para workflows baseados em tarefas e de GAP para workflows baseado em serviços. No primeiro cenário, a interface GAT pode utilizar GRMS (GRMS, [S.d.]) para executar a seleção de recursos em tempo de execução. Através do uso de GAP, Triana fornece *bindings* para Web, WSRF e P2P.

3.1.6 Askalon

Askalon é um projeto desenvolvido pela Universidade de Innsbruck na Áustria que tem como objetivo construir uma arquitetura para suportar a execução de aplicações científicas em ambiente de grid. A arquitetura Askalon (FAHRINGER *et al.*, 2007) mostra-se abrangente, contendo componentes que suportam a definição de workflows, gerência e escalonamento de recursos, máquina de execução de workflows, monitoramento e predição de desempenho. As soluções desenvolvidas para o ambiente Askalon são fortemente baseadas no *mediador* Globus (GT4) (GLOBUS, [S.d.]).

O escalonador proposto por Askalon constitui um serviço que prepara a aplicação do workflow para a execução no grid. O workflow, descrito em AGWL (*Abstract Grid Workflow Language*), é processado para ser convertido em uma forma executável e escalonado para recursos do grid. A estratégia de execução baseia-se em dois componentes principais: *Workflow Converter* e *Scheduling Engine*.

Workflow Converter é o componente responsável por promover simplificações no workflow definido pelo cientista. O componente *Scheduling Engine* é responsável pelo mapeamento de recursos propriamente dito, sendo baseado numa arquitetura plugável, a qual permite o uso de diferentes algoritmos para a produção do escalonamento final. Em sua versão padrão, o componente *Scheduling Engine* aplica o algoritmo HEFT (*Heterogeneous Earliest Finish Time*) (TOPCUOGLU *et al.*, 2002), o qual consiste de 3 fases: (i) criar pesos para os nós do grid baseado em tempos de execução estimados, além de definir pesos para as transferências de dados; (ii) definir valores de ranque para as tarefas do workflow à medida que o mesmo é atravessado, o valor de ranque é o tempo estimado do nó mais os tempos de seus sucessores, gerando uma lista em ordem decrescente dos valores de ranque; e (iii) cada tarefa recuperada da lista é mapeada para o recurso que forneça o menor tempo de execução.

Outro relevante componente da arquitetura Askalon é o *Performance Prediction*, o qual colabora com o *Scheduling Engine* e apresenta uma proposta interessante sobre previsão de desempenho em ambiente de grid. As principais tarefas fornecidas por esse componente são: *predição em nível de sistema*; *predição em nível de aplicação* e *predição em nível de workflow*.

A predição em nível de sistema inclui previsões de rede (largura de banda, atraso, etc.) e previsões do computador (CPU, memória, armazenamento, etc.). Em nível de aplicação são previstos os tempos de execução de atividades atômicas: aplicações e transferência de dados. Finalmente, em nível de workflolw são previstos os tempos de execução de sub-workflows no grid, baseando-se nas previsões obtidas pelo nível de aplicação. A previsão de desempenho em nível de (sub)workflow, torna-se particularmente interessante haja vista que uma das propriedades de workflows científicos é a sua re-execução.

3.1.7 GridAnt

GridAnt (AMIN *et al.*, 2004) trata-se de um sistema de gerência de workflows baseado na conhecida ferramenta Ant (APACHE, [S.d.]). Tradicionalmente, Ant provê um mecanismo flexível para expressar as dependências de script na construção de projetos. Em GridAnt, a ferramenta Ant é usada como máquina de execução dos workflows, gerenciando as dependências entre as tarefas que podem executar no grid. Para tal, Ant é estendida para prover novas funcionalidades que permitam a composição de workflows, garantindo a comunicação entre tarefas.

A arquitetura GridAnt é composta de quatro componentes principais: máquina de execução de workflows; ambiente de tempo de execução; vocabulário de workflow; e monitor de workflow.

O ambiente de tempo de execução garante a troca de dados entre tarefas do workflow através da implementação de um modelo de comunicação, globalmente acessível, no estilo *whiteboard*. Nesse ambiente é possível manipular estruturas de dados que podem ser lidas ou escritas por tarefas individualmente. As tarefas dos workflows GridAnt são descritas em XML, similar à descrição em Ant, sendo cada uma delas associada a uma tag. Para ser executada, uma tarefa precisa esperar pela notificação de todas as tarefas das quais ela dependa. Um conjunto de tarefas em GridAnt pode ser executado de duas formas: sequencial e paralela. Exemplos de tarefas são *grid-copy* e *grid-delete*, as quais são voltadas para a gerência de dados, significando a cópia e remoção de arquivos entre recursos do grid, respectivamente.

GridAnt não fornece nenhuma estratégia mais sofisticada para promover a execução de seus workflows, limitando-se ao uso das funcionalidades providas pelo mediador de grid Globus. Assim, GridAnt dispõe de uma tarefa denominada *grid-execute*, a qual é responsável pela execução dos workflows nos recursos do grid através de chamadas ao Globus.

3.1.8 ICENI

Imperial College e-Science Network Infrastructure (ICENI) (MCGOUGH *et al.*, 2004) é um mediador de grid desenvolvido pelo London e-Science Centre. ICENI faz uso da linguagem XML para fazer a representação de worflows abstratos. Em ICENI, workflows descrevem uma coleção de componentes e links entre eles, e são denominados *Execution Plans* (EP).

Cada componente do workflow é descrito em função de seu significado, fluxo de controle e implementação. O significado de um componente representa a descrição de alto nível da função que o componente desempenha. O fluxo de controle indica o comportamento desse componente. A implementação do componente indica o algoritmo utilizado pelo componente para atender a sua proposta. Seguindo esse modelo, o significado de um componente pode ter múltiplos comportamentos e para cada comportamento podem existir diversas implementações diferentes.

ICENI provê diversas estratégias de escalonamento para mapear o workflow abstrato em recursos do grid. Uma das estratégias de escalonamento faz uso de um repositório de dados de desempenho, o qual tem por objetivo reunir informações sobre o desempenho de execuções prévias dos componentes a fim de utilizar essas informações para estimar o tempo de execução de futuras execuções de cada componente pertencente ao workflow. O repositório armazena ainda metadados, tais como: qual recurso em que o componente foi executado; qual a implementação usada pelo componente; e o número de outros componentes concorrentemente executando no mesmo recurso.

3.1.9 GWFE

Gridbus Workflow Engine (GWFE) (YU, BUYYA, 2004) é um sistema de código aberto desenvolvido pela Universidade de Melbourne no âmbito do projeto Gridbus (GRIDBUS, [S.d.]) (BUYYA, VENUGOPAL, 2004) cujo principal objetivo é oferecer suporte à execução de workflows em grids. Apesar de estar integrado ao Gridbus Broker, GWFE pode ser utilizado com outros mediadores de grid (ex. Globus).

GWFE possui uma linguagem baseada em XML para a definição de workflows na qual o projetista do workflow pode definir as tarefas e suas dependências. A arquitetura do sistema de gerência de workflows é dirigida pelos requisitos do conceito de Economia de Grid (BUYYA *et al.*, 2001). O mecanismo de Economia de Grid apresenta uma técnica para a gerência de recursos do grid baseada na regulação entre fornecimento e demanda de recursos.

GWFE possui um sistema de escalonamento descentralizado e hierárquico proposto com o intuito de lidar com a heterogeneidade e dinamismo de ambientes de grid. Para tal, a técnica de escalonamento suporta o planejamento *just in-time*, sendo dirigido a eventos e atuando como instância única para cada tarefa do workflow. Ou seja, cada tarefa possui seu próprio escalonador (*Task Manager*), o qual implementa um algoritmo de escalonamento e trata do processamento da tarefa (seleção e negociação de

recursos, *dispatcher* e tratamento de falhas). *Task Managers* (TMs) são criados e controlados pelo *Workflow Coordinator* (WCO) e comunicam-se através de um serviço de evento. A comunicação entre WCO e TMs garante que sejam preservadas as dependências entre tarefas, haja vista que TMs operam como escalonadores independentes (visão descentralizada) e por conta disso, podem ser executados em paralelo. A notificação dos eventos gerados na comunicação baseia-se em um modelo de subscrição/notificação suportado através da implementação de um espaço de tuplas.

3.2 **A Computação Verde aplicada ao Escalonamento**

Na seção 3.1 foram apresentadas as principais características dos SGWfCs. A partir dessa análise, não foi encontrada em nenhum SGWfC a implementação de técnicas de escalonamento baseadas em políticas de computação verde. De fato, boa parte dos sistemas avaliados encontra-se preocupada com as características tradicionais do escalonamento de recursos em grid tais como: *tempo de execução*; *balanceamento de carga*; *largura de banda* e *requisitos do sistema/usuário*.

Apesar da ausência de soluções de computação verde em SGWfCs, sabe-se que muitos deles são construídos sobre uma infra-estrutura básica de grid fornecida pelos mediadores (globus, gLite, etc.). Tal realidade viabiliza a aplicação da computação verde a partir da gerência de recursos exercida pelos mediadores. O mediador gLite, por exemplo, tem sido utilizado pela plataforma EGI (*European Grid Infrastructure*) (EGI, [S.d.]) a qual mantém um grid com mais de 260 sítios espalhados entre 55 países. Dentre os experimentos que utilizam EGI, encontra-se o WLCG (Worldwide LHC Computing Grid) (WLCG, 2002), responsável pela plataforma computacional do projeto LHC (LHC, [S.d.]).

O mediador gLite possui um sistema de gerência de carga de trabalho denominado *Workload Management System* (WMS), o qual executa na máquina *Resource Broker* (RB). RB é responsável por fazer a análise dos recursos disponíveis do grid e selecionar aquele que executará a próxima tarefa. A escolha do recurso é feita através da técnica denominada casamento (*matching*), conforme mencionada na seção 3.1. A técnica consiste em selecionar dentre todos os recursos disponíveis do grid aqueles que preencham os requisitos do usuário e os especificados nos arquivos de entrada. A partir disso, o RB escolhe o recurso melhor ranqueado segundo informações

do status do recurso (tipicamente uma função do número de jobs em execução sobre o número de jobs na fila).

Dessa maneira, pode-se concluir que a técnica de casamento atualmente aplicada nos grids que utilizam gLite pode ser facilmente adaptada para a inclusão de restrições de computação verde, através de uma possível análise das características de consumo de energia dos recursos do grid.

3.2.1 Análise do Escalonamento nos SGWfCs Avaliados

Esta seção traz uma análise comparativa do processo de escalonamento realizado nos SGWfCs estudados. A tabela 1 reúne as principais características dos escalonadores implementados nesses ambientes. Conforme já havia sido mencionado anteriormente, nenhum dos SGWfCs faz uso de algum critério de computação verde. Em vez disso, boa parte dos trabalhos baseiam suas soluções de escalonamento em critérios tradicionais tais como estimativas de tempo de execução e custo.

Tabela 1: Características do escalonamento nos SGWfCs

SGWfC	Estratégia Aplicada	Integração com Grid	Critérios	Algoritmo de Escalonamento
Taverna	Definida pelo usuário	Serviços web, Soaplab	N/A*	N/A
Kepler	Definida pelo usuário / Baseada nos tipos de diretores	Serviços web	N/A	N/A
Vistrails	Gerente de Cache	Serviços web	N/A	N/A
Pegasus	Definido pelo usuário / Baseada em técnicas de particionamento de workflows	Globus Toolkit, Condor	Tempo de Execução	Min-Min, Grasp
Triana	Baseada em GAT	GRMS/ GAT(JXTA, serviços web)	Tempo de Execução	N/A
Askalon	Baseada em simplificações de workflows e algoritmos	Globus Toolkit	Tempo de Execução	HEFT / Permite a inclusão de outros algoritmos
GridAnt	Baseada em ambiente Ant adaptado para funcionar com Globus	Globus Toolkit	N/A	N/A
ICENI	Utiliza uma base com dados de execuções anteriores	Jini, JXTA e Globus Toolkit	Tempo de Execução e Custo	Permite plugar novos algoritmos
GWFE	Hierárquica e descentralizada, cada tarefa possui seu próprio escalonador	Gridbus Broker, Globus Toolkit	Tempo de Execução e Custo	Regulação entre fornecimento e demanda de recursos

* N/A: Não se Aplica

Taverna, Kepler e Vistrails diferenciam-se dos demais SGWfCs por não atuarem diretamente sobre os recursos do grid. Isso ocorre devido à integração dos mesmos com o ambiente de grid, a qual acontece de forma indireta, através da invocação de serviços

web. O que os distanciam do controle exercido sobre os recursos do grid, dificultando a implementação de estratégias de escalonamento mais elaboradas e deixando a escolha dos recursos a cargo do usuário.

Em Triana, tem sido desenvolvido um sistema que busca operar de forma independente de mediadores. O escalonamento e a gerência de recursos do grid baseiam-se no uso da plataforma GAT e GRMS, não deixando clara a existência de um algoritmo mais específico para o escalonamento. GridAnt também não oferece uma estratégia de escalonamento diferenciada, mantendo apenas as funcionalidades de execução fornecidas pela ferramenta Ant adaptada para as funcionalidades do ambiente de grid.

Pegasus, Askalon, ICENI e GWFE colocam-se próximos à infraestrutura de grid, assumindo o controle de recursos através do uso imediato de mediadores. Pegasus possui a flexibilidade de permitir que o próprio cientista escolha os recursos em que as tarefas irão executar. Além disso, Pegasus também implementa algumas estratégias de escalonamento tais como Min-Min e Grasp. A primeira provê um escalonamento para cada tarefa individualmente, enquanto a última fornece um escalonamento para o workflow inteiro, ambas visando minimizar o tempo de execução.

Similarmente ao Pegasus, GWFE possui um escalonador para cada tarefa do workflow, tendo sido construído sobre uma visão hierárquica e descentralizada. Outra característica que o distingue dos demais SGWfCs é o fato de utilizar um modelo de economia que impõe um equilíbrio entre o fornecimento e a demanda por recursos do grid. Os critérios analisados por GWFE correspondem a estimativas de tempo de execução e custo pelo uso dos recursos, não avaliando temas da computação verde tal como o consumo de energia.

Askalon e ICENI implementam algoritmos de escalonamento voltados para a minimização do tempo de execução do workflow, entretanto, ambos dispõem de mecanismos que permitem a inclusão de novos algoritmos de escalonamento. Tal característica os torna mais propícios para a aplicação de políticas de computação verde.

3.2.2 Escalonamento com Computação Verde em HPC

Em (BUYYA *et al.*, 2010) e (BELOGLAZOV, BUYYA, 2010) é proposta uma estratégia para promover a eficiência energética em nuvem através da realocação dinâmica de máquinas virtuais. O trabalho consiste em prover escalonamento dinâmico para a execução das máquinas virtuais dos usuários com o objetivo de minimizar o

consumo de energia da nuvem. Desta forma, são implementadas heurísticas que norteiam a migração das máquinas virtuais a partir das seguintes estratégias: (i) quando a utilização do recurso está próxima de 100%, deve-se migrar para garantir o que foi negociado com o usuário; (ii) quando a utilização do recurso está baixa, deve-se migrar todas a máquinas virtuais para outro nó e o recurso deve ser desligado; (iii) quando uma máquina virtual faz acesso intenso a outra máquina virtual remota, deve-se fazer a migração para manter ambas no mesmo recurso; (iv) quando a temperatura de um recurso exceder determinado limite, deve-se migrar suas máquinas virtuais para que o mesmo possa ser resfriado sem sobrecarregar o sistema de refrigeração. Além disso, a proposta também inclui uma possível reconfiguração das máquinas ligadas mediante o monitoramento da carga de trabalho, podendo colocar as máquinas físicas em estado de baixo consumo e, possivelmente, para operar em níveis de baixo desempenho, utilizando a técnica de DVS.

Grid5000 (CAPPELLO *et al.*, 2005) representa um exemplo de infraestrutura de grid que utiliza máquinas virtuais associadas ao conceito de reserva antecipada. Em (ORGERIE *et al.*, 2008a), é apresentada uma proposta para economia de energia na plataforma Grid5000. A proposta utiliza um modelo de estimativa de consumo de energia baseado em algoritmos de predição (ORGERIE *et al.*, 2008b) que analisam um histórico de reservas anteriores. A análise das reservas permitiu aos autores concluírem que alguns recursos possuem alto consumo de energia nos períodos de inicialização. A partir disso, foi proposto um alocador de recursos que se baseia nas seguintes estratégias: desligar recursos pouco utilizados; manter o recurso ligado quando for previsto que o mesmo virá a ser utilizado novamente em um futuro próximo; e agregar reservas no mesmo recurso para evitar ciclos de liga/desliga.

O algoritmo de escalonamento HAMA (Heterogeneity Aware Meta-Scheduling Algorithm) (GARG, BUYYA, 2009) descreve uma estratégia para a redução de energia em escalonadores de grid. HAMA está apoiado em um modelo de energia que considera a eficiência energética do processador e do sistema de refrigeração. O processo de alocação tem início a partir da ordenação das tarefas por prazo final (estabelecido pelo usuário) e dos recursos do grid de acordo com sua eficiência energética, calculada segundo o modelo de energia. A estratégia consiste em alocar a tarefa mais urgente (de prazo final mais cedo) para executar no recurso mais eficiente energeticamente. Para cada recurso é definido um intervalo de frequência de CPU $[f_{min}, f_{max}]$ em que o mesmo deverá operar. O escalonador então designa uma fatia de tempo para que as tarefas

utilizem os recursos com frequência mínima (economizando o máximo de energia). Entretanto, a qualquer momento o valor da frequência pode ser aumentado (reestabelecendo o melhor desempenho do recurso) em face de ameaça ao cumprimento do prazo final estabelecido pelo usuário. Desta forma, HAMA busca explorar a heterogeneidade dos recursos aplicando a técnica DVS para permitir que as tarefas sejam executadas tanto quanto possível em um recurso com frequência mínima, consumindo o mínimo de energia.

Além dos trabalhos apresentados anteriormente (exceto Grid5000), muitos outros trabalhos em HPC (HOTTA *et al.*, 2006) (JEJURIKAR, GUPTA, 2006) (LEPING, YING, 2008) (KYONG *et al.*, 2007) (MEISNER *et al.*, 2009) (TESAURO *et al.*, 2007) (VERMA *et al.*, 2008) (KAPPIAH *et al.*, 2005) (HSU, FENG, 2005) (LAMMIE *et al.*, 2009) aplicam políticas de computação verde através da utilização da técnica DVS. Trata-se de uma técnica que permite a redução imediata do consumo de energia da CPU. Todavia, traz consigo a desvantagem de ser intrusiva sob o ponto de vista do escalonador global (meta-escalonador) em ambiente de grid. De fato, torna-se inviável aplicar DVS em um grid nos moldes do WLCG (WLCG, 2002) pelo fato de ferir a autonomia dos sítios que colaboram com o projeto. Outro aspecto negativo do uso da técnica DVS diz respeito ao fato de que a mesma ainda não se encontra disponível para todas as CPUs existentes. Modelos de CPU mais antigos que não permitam a redução de voltagem/frequência podem eventualmente compor o conjunto de recursos computacionais de algum sítio pertencente ao grid.

A manipulação de máquinas virtuais constitui outra característica dissonante do ambiente de grid aqui considerado. De fato, máquinas virtuais configuradas por usuários para representar e executar suas tarefas não faz parte da infraestrutura de grid tida como referência deste trabalho, conforme será explicado no Capítulo 5. Tal característica se faz presente em ambientes como nuvem e grids utilitários. Outro elemento que não se encontra disponível no modelo de grid aqui estudado é o mecanismo que permite aos usuários fazerem reservas antecipadas para uso do ambiente. Esse mecanismo serve de base para a solução verde apresentada em Grid5000, tal fato inviabiliza sua aplicabilidade no cenário avaliado neste trabalho.

Diante das considerações apresentadas neste capítulo, fica clara a existência de um forte e recente movimento da comunidade científica na direção de construir soluções de computação verde para ambientes HPC. Apesar disso, SGWfCs integrados à tecnologia de grid ainda não estão atendendo a essa demanda. Tal realidade, abre uma

excelente oportunidade para a investigação da aplicabilidade da computação verde em grids de escala global que executam geograficamente distribuídos, similares ao modelo empregado pelo WLCG.

O capítulo seguinte discute os conceitos envolvidos na proposição de um modelo para a representação do consumo de energia dos recursos do grid.

Capítulo 4 – Modelo de Dados

A utilização de *benchmarks* tem sido uma prática comum na avaliação de equipamentos em ambientes HPC. De fato, diversas entidades e associações têm trabalhado para fornecer medidas que permitam uma análise comparativa dos recursos computacionais. O modelo de dados descrito neste capítulo utiliza métricas obtidas a partir da execução prévia de alguns *benchmarks*. O uso de resultados de *benchmarks* pelo escalonador coloca-se como uma alternativa interessante devido às seguintes razões:

- Precisão e confiabilidade dos dados calculados.
- Fornece uma infraestrutura uniforme com métodos documentados que podem ser empregados em todos os sítios do grid.
- Acesso público – geralmente, os programas dos *benchmarks* são software livre.
- Característica multi-plataforma – podem ser executados em diferentes ambientes, transpondo a heterogeneidade dos recursos do grid.

Este capítulo descreve um modelo de energia para representar as características relacionadas ao consumo de energia dos recursos do grid para a execução de experimentos científicos. As seções subsequentes (4.1 a 4.3) discutem os conceitos envolvidos no modelo e, finalmente, a última seção (4.4) apresenta o modelo estruturado em um diagrama de classes.

4.1 Tarefas

Conforme descrito na seção 2.1, grids computacionais de escala global costumam ter usuários estruturados em VOs (*Virtual Organizations*). Normalmente, cada VO está voltada para um domínio exclusivo (física de altas energias, bioinformática, etc.), oferecendo suporte a um conjunto de aplicações que interessam ao grupo de usuários da VO em questão. Um dos pontos explorados pelas estratégias de escalonamento propostas neste trabalho prevê a avaliação dos perfis das aplicações lançadas pelas tarefas.

Grupos de e-ciência têm se preocupado com a produção de *benchmarks* voltados para seus experimentos. Para a área de física de altas energias, por exemplo, vem sendo

desenvolvido o *benchmark* HEP-SPEC06 (SULI, 2008) (HEP-SPEC06, [S.d.]), o qual busca adaptar o *benchmark*, de uso geral, SPEC CPU2006 (CPU2006, [S.d.]) às necessidades da física de altas energias. Outra iniciativa que ratifica o interesse na criação de *benchmarks* vem da área de bioinformática. Os *benchmarks* BioBench (ALBAYRAKTAROGLU *et al.*, 2005) e BioPerf (BADER *et al.*, 2005) disponibilizam métodos eficientes para a análise do desempenho de aplicações de bioinformática frente a diferentes arquiteturas de computador.

Desta forma, as métricas fornecidas pelos *benchmarks* podem auxiliar o escalonador na tomada de decisão sobre qual recurso do grid alocar para uma possível tarefa a ser executada. Alguns detalhes da arquitetura do recurso computacional (ex. a quantidade de memória cache) podem trazer mais benefício para a execução de uma aplicação do que outra. E, consequentemente, contribuir para a redução do *makespan*.

Algumas informações acerca das tarefas são relevantes para o modelo utilizado pelo escalonador. Essas informações permitem a caracterização das tarefas em perfis, de modo que as mesmas possam ser avaliadas antes do início da execução. Para perfilar as tarefas, o escalonador precisa obter informações sobre: a complexidade da tarefa; sua característica de execução; e seu domínio de conhecimento. A complexidade de uma tarefa pode ser obtida através de informações de execuções anteriores, modelos de estimativa ou métricas mais simples como MI (Milhões de Instruções). A característica de execução de uma aplicação pode indicar o quanto ela faz acesso a dados (aplicação de dados intensivos) ou a CPU (aplicação de CPU intensiva). Essa informação também pode ser obtida a partir de execuções anteriores juntamente com informações de dados de entrada. Por fim, o domínio de conhecimento da tarefa irá permitir a correta análise dos recursos mediante as métricas fornecidas pelos *benchmarks* correspondentes.

4.2 *Bag-of-Tasks*

No modelo aqui considerado, o meta-escalonador recebe como entrada um lote de tarefas do cientista e deverá produzir como saída, uma lista dos recursos computacionais designados para executar tais tarefas. Sendo assim, cada lote pode ser visto como um *bag-of-tasks*, ou seja, um grupo de tarefas livres de dependência que podem ser executadas em qualquer ordem.

As aplicações *bag-of-tasks* resolvem vários problemas relevantes tais como análise genômica, mineração de dados, pesquisa massiva, manipulação de imagens e

simulações Monte Carlo. Esta última corresponde às produções de Monte Carlo (*MC Productions*) submetidas ao grid WLCG do projeto LHC.

4.3 Recurso Computacional

Conforme discutido na seção 2.3, a computação verde pode ser aplicada a partir de diferentes vertentes. Em ambientes HPC, os trabalhos encontrados na literatura colocam a redução do consumo de energia como principal demanda da área. Seguindo essa linha, o escalonador proposto neste trabalho busca explorar a heterogeneidade dos recursos do grid para promover a redução do consumo de energia do grid. Um grid de escala global pressupõe a existência de diversos CPDs espalhados por diversos sítios participantes.

A energia consumida em ambientes de CPD abrange componentes de iluminação, refrigeração, redes e computação. Neste trabalho, a redução do consumo de energia passa apenas pelos elementos computacionais, excluindo inclusive os componentes de redes. Desta forma, o modelo deve ser capaz de capturar apenas informações referentes ao consumo de energia dos recursos computacionais.

Um computador é constituído de diversos componentes que consomem energia em seu funcionamento: memória, placa-mãe, processador, placa de vídeo, disco, etc. Neste trabalho, o foco tem sido direcionado apenas para o consumo da energia dispendida em processamento e armazenamento. Sendo assim, o escalonador precisa dispor de informações sobre consumo e eficiência energética para *recursos de execução* e para *recursos de armazenamento*. As seções seguintes apresentam, em detalhe, as informações capturadas pelo modelo para medir a eficiência energética nesses dois ambientes.

4.3.1 Consumo Energético em Recursos de Armazenamento

A decisão sobre qual recurso de armazenamento do grid escolher para acessar ou gravar grandes volumes de dados pode influenciar de forma decisiva no consumo de energia global do grid. De fato, o consumo de energia com discos tem impacto significativo, principalmente, quando considerada a alocação de aplicações de *dados intensivos*. Partindo desse princípio, em recursos de armazenamento a análise da eficiência energética é dominada apenas pelo consumo de energia dos discos que compõem o subsistema da máquina, sendo ignorado o consumo energético dos demais componentes.

Para aferir a eficiência energética dos discos dos recursos de armazenamento do grid foi empregada a métrica IOPS (*Input/Output Operations Per Second*), uma medida bastante utilizada para medir o desempenho de HDs, combinada à taxa de consumo de energia dos discos (em Watts). Uma possível estratégia para a obtenção do valor de IOPS dar-se através da execução do *benchmark* IOzone (IOZONE, [S.d.]) sobre recursos de armazenamento de cada sítio do grid. Em seguida, seria calculada a medida de eficiência energética em *IOPS/Watt*, através do valor de IOPS sobre a taxa de consumo energético do disco.

Portanto, o modelo aqui proposto prevê que os sítios do grid publiquem as taxas de IOPS/Watt para cada um de seus recursos de armazenamento.

4.3.2 Consumo Energético em Recursos de Execução

O modelo de dados descrito neste capítulo permite a análise da eficiência energética dos recursos de execução através de resultados de *benchmarks* especializados nesse intuito. Diferentemente da análise feita em recursos de armazenamento, para recursos de execução os *benchmarks* utilizados verificam o consumo de energia do subsistema como um todo, e não apenas de seus processadores. Assim, de acordo com a estratégia aplicada, o escalonador poderá lançar mão dos seguintes *benchmarks*: SPECpower (SPECPOWER, [S.d.]), TPC-Energy (TPC-ENERGY, [S.d.]) e Green500 (GREEN500, [S.d.]) (SHARMA *et al.*, 2006).

SPECpower (também conhecido como SPECpower_ssj2008) é um *benchmark* desenvolvido por uma organização sem fins lucrativos denominada SPEC (*Standard Performance Evaluation Corporation*). O objetivo desse *benchmark* é medir o consumo de energia de um computador em relação a seu desempenho. Para tal, SPECpower dispõe de uma solução desenvolvida em java (SSJ *Workload*) que funciona em duas fases: Calibragem e Execução em Séries de Cargas de Trabalho. Na fase de Calibragem, o programa determina o *throughput* máximo que o subsistema avaliado pode sustentar. Sendo assim, o valor do *throughput* obtido passa a ser usado como referência para determinar os níveis de carga de trabalho. Ou seja, durante a fase de execução, os níveis de carga são estabelecidos como uma porcentagem do *throughput*, sendo decrescentemente variada a carga de trabalho de 100% a 0% em intervalos de 10. Para cada nível, o programa registra o número de operações *ssj_ops* executadas (transações dos seguintes tipos: *novo pedido*, *pagamento*, *status do pedido*, *entrega*, *verificação de*

estoque e *relatório do cliente*) e a média do consumo de energia da máquina em watts. Assim, SPECpower gera uma métrica correspondente a *ssj_ops/watts*.

É importante destacar um aspecto positivo da solução SPECpower, trata-se do fato de que a mesma não se baseia apenas em medições de máquinas com carga máxima. Sendo assim, a estratégia de medição adotada em SPECpower evita a aferição de possíveis idiossincrasias do consumo energético da máquina estressada.

TPC-Energy corresponde a uma especificação produzida pela TPC (*Transaction Processing Performance Council*), uma organização sem fins lucrativos especializada na construção de *benchmarks* para processamento de transações e banco de dados. O objetivo dessa especificação é prover métricas relacionadas ao consumo de energia para os *benchmarks* da TPC. Ao longo dos anos, TPC tem desenvolvido uma série de *benchmarks* tais como: TPC-C (*benchmark* OLTP), TPC-E (*benchmark* OLTP que simula uma carga de trabalho de uma empresa de corretagem) e TPC-H (*benchmark* de apoio à decisão). Para facilitar a implementação das métricas de consumo de energia, TPC tem desenvolvido um software denominado EMS (*Energy Measuring System*). EMS é responsável pela comunicação com equipamentos e sensores de medição de energia. Ele também faz a interface com *drivers* para a execução dos *benchmarks* TPC. As métricas de energia obtidas pela aplicação da especificação TPC-Energy são derivadas a partir de métricas de desempenho resultantes dos *benchmarks* TPC. Sendo assim, a métrica de energia gerada por um *benchmark* TPC seria calculada a partir da divisão do consumo de energia (em watts) por unidade de trabalho (por exemplo: transações, consultas). Considerando o *benchmark* TPC-C, o resultado da métrica de energia seria dado em Watts/KtpmC, onde tpmC corresponde ao número de transações (de novo pedido) por minuto.

Outra medida de eficiência energética utilizada pelo modelo adotado neste trabalho vem da lista Green500. Conforme afirmado na seção 2.4, Green500 baseia-se nos resultados da lista TOP500, os quais são ranqueados através do *benchmark* LINPACK (DONGARRA *et al.*, 2003). LINPACK corresponde a uma biblioteca de software voltada para a execução de álgebra linear numérica em computadores. Tal execução ocorre através do uso de um conjunto de bibliotecas denominado BLAS (*Basic Linear Algebra Subprograms*) (LAWSON *et al.*, 1979). A execução do pacote LINPACK fornece a métrica para medição de desempenho FLOPS (*Float Operation Per Second*). Green500 executa o *benchmark* LINPACK fazendo a aferição do consumo de energia em watts. Sendo assim, Green500 fornece como métrica de eficiência

energética o *throughput* em FLOPS para cada watt de energia consumido, comumente da seguinte forma: MFLOPS/W.

4.4 Análise do Modelo

Esta seção reúne as informações e conceitos discutidos nas seções anteriores, estruturando-os em um modelo de dados representado em diagrama de classes UML, conforme mostra a Figura 1. Desta forma, o modelo ajuda a entender os relacionamentos entre os conceitos e métricas abordados anteriormente.

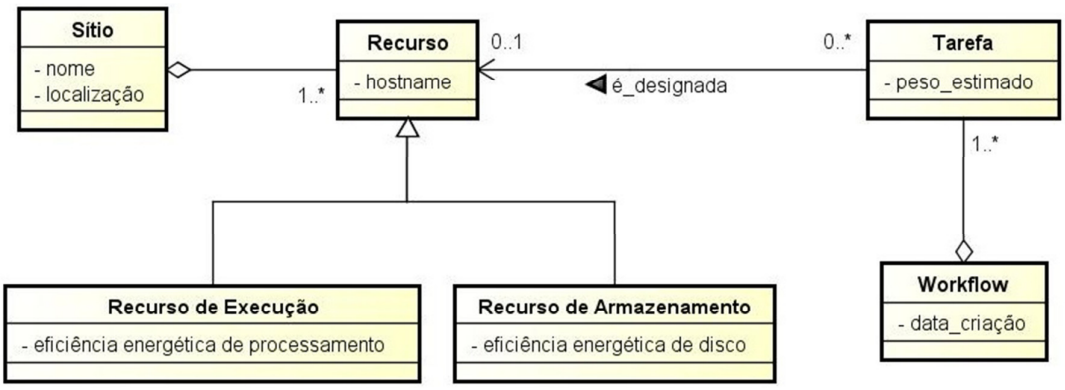

Figura 1: Modelo de dados

De acordo com o modelo, os sítios do grid possuem recursos que podem ser dos tipos: execução e armazenamento. Para ambos os tipos, o escalonador possui informações que correspondem aos graus de eficiência energética dos recursos, os quais são obtidos conforme discutido na seção 4.3. De acordo com a estratégia adotada pelo escalonador e em face da análise de seus perfis, cada tarefa é designada para executar unicamente em um dado recurso.

Capítulo 5 – Arquitetura

Este capítulo descreve o funcionamento do grid considerado neste trabalho e apresenta os componentes adicionados pela solução de escalonamento proposta. Para entender melhor o funcionamento do escalonador é necessário primeiro entender como funciona o grid. A seção 5.1 explica em detalhes o funcionamento do tipo de grid explorado neste trabalho enquanto a seção 5.2 apresenta uma visão geral da arquitetura e seus componentes.

5.1 Infraestrutura e Funcionamento do Grid

Conforme discutido no capítulo 2, existem diversos tipos distintos de grids. O modelo de grid considerado neste trabalho possui como referência o WLCG – grid do projeto LHC. Desta forma, considera-se que um grid é composto de diversos sítios geograficamente distribuídos que fornecem recursos para a execução das tarefas do grid. Normalmente, essas tarefas compõem workflows ou lotes de produção.

O número de tarefas em execução no grid depende em muito do funcionamento do experimento ou detector. Ou seja, durante períodos de produção, os detectores estão em funcionamento pleno, realizando a coleta de dados. Os dados coletados vão para a análise *in-silico* (nos computadores), sendo submetidos ao grid computacional como um conjunto ou lote de tarefas.

À medida que as tarefas são submetidas para a execução, cada uma delas precisa ser destinada ao sítio em que será executada, dando início ao processo de escalonamento. Pode-se afirmar que o escalonamento de um lote de tarefas neste modelo de grid ocorre em dois níveis: global (no âmbito do grid) e local (no âmbito do sítio).

Essa visão hierárquica do escalonamento explica-se através da própria infraestrutura do grid. Cada sítio do grid executa um sistema de gerência de filas (por exemplo: Torque PBS) associado a um escalonador de *jobs* (por exemplo: maui), normalmente, este último baseia suas escolhas em filas de prioridades. O conjunto instalado (PBS com escalonador) em um sítio é responsável pela execução das tarefas do grid (geradas pelos experimentos) que chegam nesse sítio, bem como das tarefas geradas no próprio sítio. Entretanto, antes da tarefa ser submetida ao escalonador do sítio (local), ela precisa passar pelo escalonador do grid (global). Este último irá decidir

para qual sítio a tarefa será designada. Ou seja, em um primeiro momento, o escalonador global escolhe para qual recurso do sítio a tarefa será destinada. Como já foi dito, essa escolha considera aspectos do grid tais como balanceamento de carga e *throughput*; e também considera aspectos do sítio. Chegando ao recurso escolhido, a tarefa do grid é colocada em uma ordem de execução definida pelo escalonador local, considerando também possíveis tarefas geradas no próprio sítio.

A infraestrutura e o comportamento do escalonador demonstram a característica autônoma dos sítios que operam no grid, ratificando assim, a dificuldade de uma possível aplicação da técnica de DVS, conforme discutido na seção 2.4.

Dentre os níveis de escalonamento citados anteriormente, o que ocorre no âmbito global é objeto de interesse deste trabalho. Sendo assim, as estratégias de alocação apresentadas no capítulo 6 concentram-se em determinar em que recursos e sítios as tarefas submetidas ao grid irão executar. O escalonamento em nível global do grid também é conhecido na literatura como meta-escalonamento.

A partir da compreensão do funcionamento do grid, algumas conclusões podem ser tiradas:

(i) Os recursos usados pelo grid são "dedicados", não no sentido de exclusividade ao grid, mas pelo fato das máquinas estarem sempre disponíveis (ligadas);

(ii) Os períodos de produção indicam que o grid, em sua visão global, passa por momentos de baixa e alta carga de trabalho, consumindo assim, menores e maiores quantidades de energia, respectivamente.

Considerando a conclusão apontada pelo item (i), sabe-se que o grid WLCG realiza medições sobre a disponibilidade dos recursos de seus sítios (DECEMBER/2011, [S.d.]). Além disso, os administradores dos sítios são constantemente cobrados no caso em que a disponibilidade atinge um patamar abaixo do esperado. A afirmação (ii) é corroborada pelos dados sobre o número de *jobs* executados nos últimos 3 meses em um experimento do WLCG, conforme mostra o gráfico exibido na Figura 2 (DIRAC, [S.d.]).

As conclusões (i) e (ii) são resultantes de características do modelo de grid considerado neste trabalho. Tais afirmações servem de base para as estratégias de alocação que visam à redução do consumo de energia do grid.

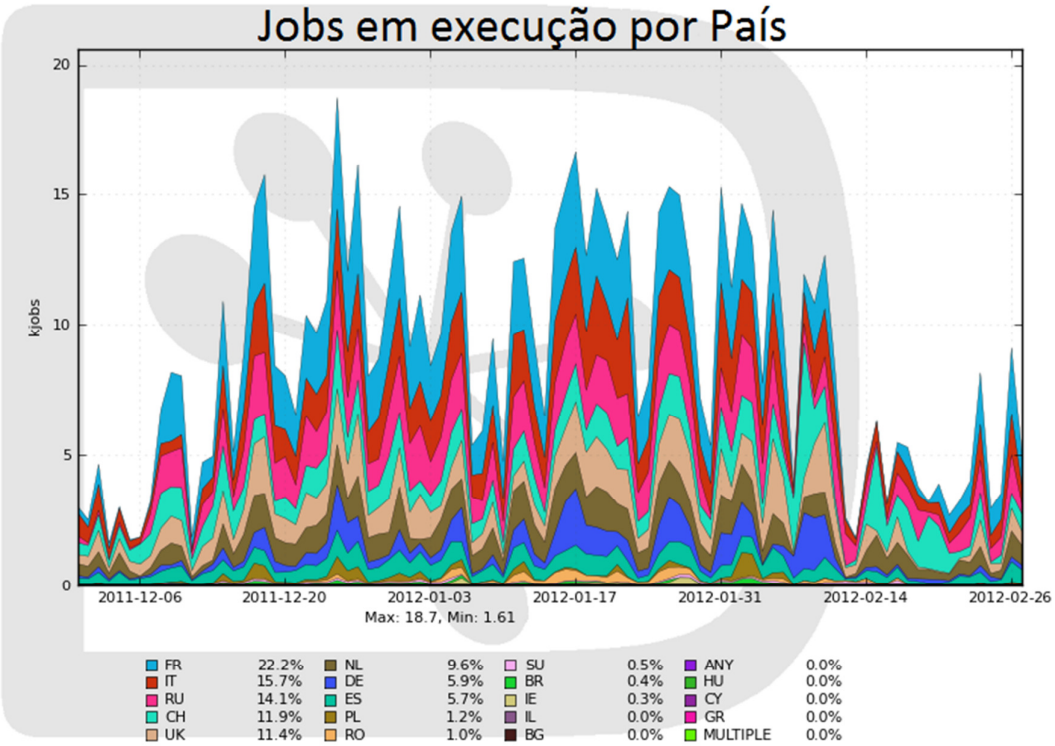

Figura 2: Número de jobs executados entre 01-12-2011 e 01-03-2012

5.2 Visão Geral da Arquitetura

Esta seção descreve os componentes que fazem parte da solução de escalonamento desenvolvida neste trabalho. A Figura 3 representa uma visão geral da arquitetura do grid incluindo os componentes envolvidos com as políticas de computação verde.

Na Figura 3 são exibidos três sítios (A, B e C), os quais representam instituições geograficamente distribuídas que colaboram entre si por meio de um grid. Tal cenário ajuda a descrever a infraestrutura do grid, onde cada sítio compartilha recursos de armazenamento (RAs) e recursos de execução (REs).

Os sítios devem ainda publicar informações sobre seus recursos, incluindo dados referentes à eficiência energética dos mesmos, conforme definido na seção 4.3. Essas informações são publicadas através do SI (Sistema de Informação) e estão disponíveis aos serviços do grid.

LT (Lote de Tarefas) representa um conjunto de tarefas submetidas ao grid através de ferramentas que compõem a parte cliente do mediador utilizado pelo grid (gLite para o WLCG). Essas tarefas são geradas a partir de usuários do grid ou advêm

de seus experimentos (os detectores ATLAS, CMS, ALICE e LHCB) conforme é mostrado na parte superior direta da Figura 3.

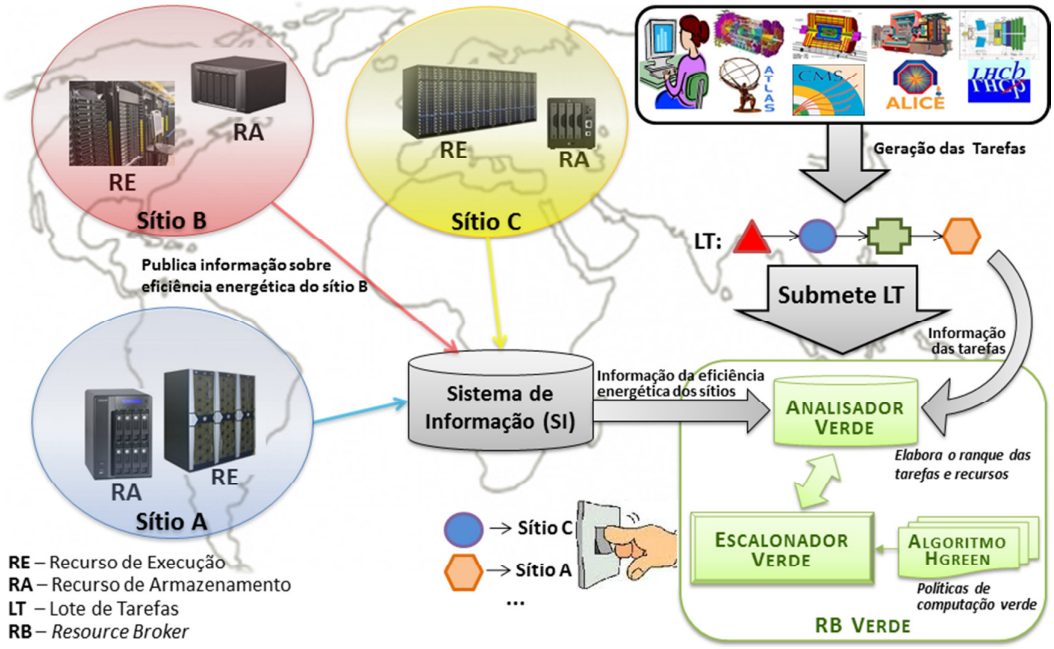

Figura 3: Visão geral da arquitetura proposta

O passo seguinte à geração das tarefas é a submissão de LT ao grid por meio de um *Resource Broker* (RB). Conforme discutido na seção 3.2, RB é o serviço responsável por fazer a análise dos recursos disponíveis do grid e selecionar aquele que executará a próxima tarefa do lote. Em se tratando da arquitetura proposta neste trabalho, tem sido proposto o RB Verde cujo objetivo é fazer a alocação dos recursos disponíveis às tarefas do lote de modo a reduzir o consumo de energia do grid como um todo. RB Verde apresenta-se na parte inferior direita da Figura 3, sendo composto de dois elementos principais: o Analisador Verde e o Escalonador Verde. Esses elementos são mais bem detalhados nas seções apresentadas a seguir.

Cabe salientar ainda que para facilitar o entendimento, a arquitetura representada pela Figura 3 possui apenas três sítios e um único RB – o RB Verde. Entretanto, considerando um grid de escala global em produção, sabe-se que a dimensão seria muito maior. No caso do grid WLCG, por exemplo, existem mais de 200 sítios em operação. Em casos desse tipo, por questões de redundância e para evitar sobrecargas, normalmente, existe mais de um RB em operação, atendendo a regiões geográficas distintas.

5.2.1 Analisador Verde

O Analisador Verde é o componente responsável por reunir informações referentes à eficiência energética dos recursos e perfis das tarefas. Conforme discutido no capítulo 4, tais informações baseiam-se em métricas obtidas a partir de *benchmarks* de domínio público.

A conduta para a obtenção das métricas estabelecidas no modelo consiste na execução, em cada sítio, dos *benchmarks* utilizados pelo escalonador. Os resultados obtidos, por sua vez, devem ser publicados pelos respectivos sítios através do SI. Dessa maneira, os RB Verdes, responsáveis pela implementação de políticas de computação verde, poderão definir a estratégia de escalonamento a ser adotada a partir da analise dos resultados dos *benchmarks*.

Conforme pode ser visto na Figura 3, o Analisador Verde não apenas coleta informações sobre recursos e tarefas, mas também é responsável por produzir ranques dos mesmos. Tais ranques correspondem a listas ordenadas de recursos e tarefas baseadas nas métricas definidas para o procedimento de execução de acordo com a estratégia escolhida pelo Escalonador Verde.

Sendo assim, o Analisador Verde disponibiliza informações que servem de entrada para o Escalonador Verde. E por outro lado, o Escalonador Verde determina o tipo de ranque de acordo com a política de computação verde adotada. Esta interação mútua entre os componentes do RB Verde é explicitada pela seta de sentido duplo presente na Figura 3.

5.2.2 Escalonador Verde

Este componente é responsável pelo processo de alocação de recursos em si. Ou seja, sua função principal é implementar os algoritmos das estratégias de escalonamento propostas neste trabalho. Para cumprir tal objetivo, o Escalonador Verde acessa os dados disponibilizados pelo Analisador Verde, conforme mencionado na seção anterior.

A Figura 3 apresenta, a titulo de ilustração, o algoritmo *HGreen* como política de computação verde utilizada pelo escalonador. Entretanto, outras estratégias de escalonamento também podem ser adotadas. A arquitetura provê um esquema genérico através do qual o escalonador poderá selecionar dentre várias estratégias de escalonamento. Além disso, é possível acrescentar novas estratégias de escalonamento a qualquer momento, bastando para isso apenas estender uma classe que implementa o algoritmo. Essas estratégias são discutidas em mais detalhe no Capítulo 6.

Capítulo 6 – Estratégias de Alocação de Recursos: aplicação de políticas verdes

Ao longo deste capítulo, serão discutidas, em detalhe, as estratégias de escalonamento propostas neste trabalho. As técnicas empregadas no escalonamento possuem como base a análise dos perfis de aplicações e o estudo das políticas de computação verde. No que se refere às políticas de computação verde, o foco deste trabalho tem sido prover alternativas para reduzir o consumo de energia do grid.

Entretanto, conforme mencionado anteriormente, o escalonador do grid possui outros critérios além da computação verde. Um exemplo de outro objetivo perseguido por escalonadores de grid e que vem sendo estudado há bastante tempo é a minimização do tempo de execução das tarefas (também conhecido como *makespan*).

Inicialmente foram investigadas propostas que tinham como objetivo apenas atender à computação verde. Todavia, o fato de objetivos distintos coexistirem no escalonador leva à reflexão de que tais objetivos poderiam exercer certo nível de conflito entre si. Em consequência disso, num passo posterior à análise isolada da computação verde, foi desenvolvida uma proposta que considera dois objetivos: a redução do consumo de energia e a minimização do *makespan*.

Sendo assim, este capítulo encontra-se conforme a seguir. As seções 6.1 e 6.2 discutem heurísticas que possuem como objetivo principal a redução do consumo de energia do grid, enquanto a seção 6.3 discute uma solução que visa atender de forma equiparada a dois objetivos: reduzir o consumo de energia e minimizar o tempo de execução do lote de tarefas.

6.1 Estratégia HGreen

Conforme argumentado na seção 2.3, existem diferentes maneiras de explorar o tema da computação verde. Quando considerada a execução distribuída em ambientes de grid, ainda mais vertentes são criadas. Normalmente, grids disponibilizam processamento por meio de recursos heterogêneos, os quais possuem níveis de eficiência energética distintos.

A heterogeneidade de recursos presente em um grid de escala global representa uma boa oportunidade para a aplicação de políticas de computação verde. A heurística apresentada nesta seção tem considerado a computação verde a partir do seguinte

princípio: *a priorização do uso de recursos mais eficientes energeticamente em detrimento de recursos menos eficientes, ao longo do tempo, causará menos danos ao meio ambiente* (como resultado da consequente redução de CO_2, dentre outros fatores).

Considerando o ponto de vista supracitado, este trabalho propõe uma nova heurística para a alocação de um lote de tarefas em um ambiente de grid global. Desta forma, denomina-se HGreen (do inglês, ***H****eavier Tasks on Maximum* ***Green*** *Resource*), a heurística laconicamente descrita conforme a seguir:

- As tarefas de carga mais pesada do lote são designadas para os recursos mais eficientes energeticamente.

Tendo em vista que o consumo de energia é proporcional à demanda de processamento, a estratégia definida por HGreen busca identificar as tarefas mais pesadas (aquelas que consumirão mais energia) para enviá-las aos recursos mais eficientes energeticamente. Uma vez que os recursos do grid estão sempre disponíveis (ligados), a ideia é manter operando em maior carga aqueles mais eficientes energeticamente.

Entretanto, como além da computação verde existem outros objetivos envolvidos na alocação de recursos em grid, HGreen tem estabelecido o *fator verde* (*fv*). O valor de *fv* corresponde a uma espécie de escala que varia de 0 a 1, indicando o quanto a política verde influenciará na decisão do escalonador. Ou seja, quanto mais próximo do valor 0, menos a política verde irá influenciar a decisão do escalonador (*fv* = 0, indica que política de economia de energia deve ser desconsiderada). O mesmo vale para o oposto, ou seja, quanto mais próximo do valor 1, mais a política verde influenciará a decisão final do escalonador.

A definição de *fv* adiciona uma flexibilidade à proposta de inclusão de HGreen em escalonadores de grid de larga escala. Tal valor poderia ser estabelecido por administradores de VOs para o caso de grids comunitários ou até mesmo definidos pelos usuários que submetem lotes ao grid.

Conforme ratificado na discussão da seção 3.2.1, o tipo de heurística tratado aqui ainda não tem sido explorado pela maioria dos escalonadores de grid. A maior parte dos escalonadores de grid está voltada para a redução do tempo de execução e do custo (BRANDIC *et al.*, 2005) (SAKELLARIOU *et al.*, 2005) (YU, BUYYA, 2006). A solução implementada em HGreen é inspirada no algoritmo HEFT (TOPCUOGLU *et*

al., 2002). Todavia, em HGreen, o objetivo principal é reduzir o consumo de energia do grid.

A implementação de HGreen tem como base as informações sobre o perfilhamento das aplicações representadas pelas tarefas e as medições da eficiência energéticas dos recursos do grid. Essas informações estão detalhadas no modelo descrito no Capítulo 4 e são obtidas a partir da implementação dos componentes apresentados no Capítulo 5. A seção 6.1.1 descreve a seguir os detalhes do algoritmo que implementa HGreen.

6.1.1 Algoritmo HGreen

O algoritmo HGreen implementa a heurística homônima e pode ser dividido em três fases: análise de tarefas, priorização de tarefas e mapeamento.

Inicialmente, declarações e funções utilizadas pelo algoritmo são descritas, considerando *m* recursos disponíveis do grid para executar um lote de *n* tarefas (linhas 2-14). A seguir, durante a fase de análise, os pesos das tarefas são definidos (linhas 16-19). O peso atribuído a cada tarefa é obtido a partir da estimativa do tempo de execução da tarefa (baseado em execuções anteriores) juntamente com o tamanho da entrada. Considerando um determinado recurso, quanto maior for o peso de uma tarefa destinada ao mesmo, maior será a quantidade de energia necessária para executá-la.

Além do peso atribuído a cada tarefa, também é definido o valor normalizado do tamanho de dados de entrada (*tde*), variando de 0 (tamanho de entrada/saída irrelevante) a 1 (tamanho de entrada/saída máximo). Sendo assim, através do valor de *tde* é possível identificar as tarefas que invocam aplicações que executam muitas operações de disco. Tal fato é relevante principalmente quando considerado o consumo de energia dos recursos de armazenamento, conforme discutido na seção 4.3.1.

Na fase de priorização, as tarefas são ordenadas decrescentemente de acordo com seus pesos (linhas 21-25). Esta classificação gera a lista ET, a qual mantém em primeira posição a tarefa que demanda maior consumo de energia estimado para sua execução em um dado recurso.

A fase de mapeamento (linhas 27-40) é iniciada através da iteração sobre a lista ET. As tarefas são selecionadas uma a uma seguindo a ordem definida em ET. Para cada tarefa, são verificadas as possíveis dependências com outras tarefas. Se a tarefa selecionada da lista não depende de outras tarefas que ainda não tenham sido

completadas, a tarefa então se encontra apta para ser alocada a um recurso de execução disponível do grid.

Tabela 2: Pseudocódigo do algoritmo HGreen

Algoritmo de Alocação de Tarefas – HGreen

```
1
2     // declarações
3     T = {t1,...,tn} // lista de tarefas a ser executada
4     R = {r1,...,rm} // lista de recursos do grid disponíveis retornada pelo SI
5     ET ← {} // lista de tarefas em ordem decrescente de consumo de energia
6     peso(tk) // retorna a peso da tarefa tk
7     pos(L, criterio) // retorna a posição do novo elemento da lista L
8     tde(tk) // retorna o tamanho de dados de entrada da tarefa tk
9     eee(r) // valor do benchmark para o recurso r
10    ipc(tk,r) // qtde de instruções por ciclo considerando a tarefa tk no recurso r
11    eea(r) // taxa de iops por watt para RAs
12    maxEE=0; // recurso que obteve valor máximo de ee
13    fv // fator verde: 0 .. 1
14    rmaxEE // recurso energeticamente mais eficiente para executar a tarefa selecionada
15
16    // fase de análise
17    for each tk in T do
18       peso(tk) = SI.getEstimatedWeight(tk);
19    end for
20
21    // fase de priorização
22    for each tk in T do
23       v = pos(ET, peso(tk));
24       add(tk, ET, v); // adiciona a tarefa tk na posição v da lista ET
25    end for
26
27    // fase de mapeamento
28    while ∃t in ET do
29       tk = ET.get(0) // obtém a tarefa de maior custo energético
30       if ready(tk) // caso tk não dependa de tarefas que ainda não foram completadas
31          for each ri in R do
32             ee(ri,tk) = fv*(eee(ri) + tde(tk)*eea(ri)) + (1-fv)*ipc(ri);
33             if ( ee(ri,tk) > maxEE )
34                maxEE = ee(ri,tk);
35                rmaxEE = ri;
36          end for
37          aloca(tk,rmaxEE);   // aloca a tarefa tk para o recurso rmaxEE
38       end if
39       remove(ET, 0) // remove o primeiro item da lista ET (a tarefa já alocada)
40    end while
41
```

Após a seleção da tarefa é iniciada a seleção do recurso que será designado para executar a mesma. Sendo assim, para cada recurso disponível no grid (obtido da lista R) é calculada a eficiência de execução (*ee*) considerando a tarefa selecionada da lista ET. O valor de *ee* pode ser calculado através da equação apresentada no quadro exibido pela Figura 4.

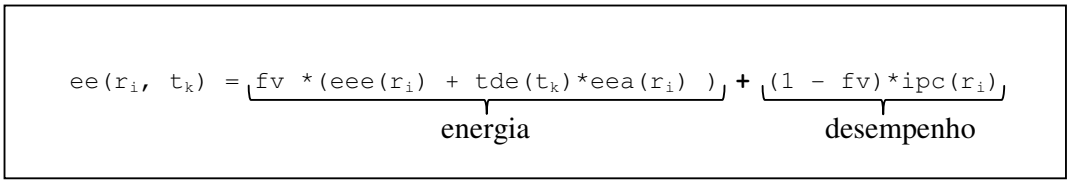

Figura 4: Fórmula para cálculo de *ee*

A fórmula $ee(r_i, t_k)$ é composta de uma soma que pode ser dividida em duas partes. A primeira parte representa a eficiência energética do recurso r_i, onde $1 \leq i \leq m$, para executar a tarefa t_k, onde $1 \leq k \leq n$. Enquanto a última parte representa o índice de desempenho de r_i em executar t_k. A seguir, os componentes da fórmula *ee* são descritos em detalhe: r_i representa o i-ésimo recurso disponível do grid; t_k representa a k-ésima tarefa do lote a ser executado; *fv* corresponde ao coeficiente de fator verde, conforme explicado anteriormente; $eee(r_i)$ representa a *eficiência energética de execução* do recurso r_i, este valor pode ser obtido a partir de alguma métrica, neste trabalho utiliza-se o *benchmark* SPECpower; $tde(t_k)$ indica o tamanho dos dados de entrada da tarefa t_k; $eea(r_i)$ representa a medida de *eficiência energética de armazenamento* do recurso r_i,: IOPS/Watts; e, finalmente, $ipc(r_i)$ corresponde à métrica de desempenho do recurso levando em conta o domínio da aplicação, por exemplo, o *benchmark* HEP-SPEC06 em se tratando de física de altas energias.

O passo seguinte é obter o recurso com valor máximo de *ee* (denominado r_{maxEE}) considerando a tarefa t_k (linhas 31-36). A seguir, o recurso selecionado r_{maxEE} é alocado para executar a tarefa t_k (linha 37). Finalmente, a tarefa é removida da lista ET (linha 39) e o mapeamento continua para a próxima tarefa selecionada na iteração seguinte. Esse processo repete-se até que todas as tarefas do lote sejam escalonadas.

A seção seguinte descreve uma estratégia gulosa para a alocação de recursos em grids globais com o objetivo de reduzir o consumo de energia independentemente dos pesos das tarefas.

6.2 Estratégia GGreen

Nesta seção discute-se uma estratégia para a alocação de recursos em grid baseada em uma solução gulosa denominada GGreen (do inglês, ***G**reedy **Green***). Diferentemente de HGreen, a estratégia GGreen não faz uso de uma única métrica que aponta a eficiência energética dos recursos tal como $eee(r_i)$. Em vez disso, são utilizadas

medições referentes aos níveis de potência dos recursos mediante o volume de carga a que estão submetidos. Essas informações também são de acesso público e igualmente fornecidas pelo *benchmark* SPECpower.

Antes de entender o funcionamento da estratégia GGreen é preciso descrever os níveis de potência atribuídos a cada recursos. Logo, seja R a lista de m recursos disponíveis do grid tal que $1 \leq i \leq m$. Para cada recurso r_i são publicados 11 diferentes níveis de potência que o mesmo poderá atingir de acordo sua carga. Sendo assim, p_{il} representa a potência atingida pelo recurso r_i quando submetido a uma carga de nível l, onde: l=0 indica carga 0 (ociosa), l=1 indica carga de 10%, l=2 indica carga de 20%, ..., l=9 indica carga de 90% e l=10 indica carga de 100% (cheia).

O nível de carga de um recurso r_i é definido pela razão entre o número de núcleos de processamento ocupados (CO_i) e o total de núcleos de processamento (C_i). Por exemplo, considere r_k um recurso com 8 núcleos. No momento em que r_k possui 4 núcleos ocupados (em execução) sua carga estará em nível l=5, ou seja, r_k estará operando com potência p_{k5}. Caso em um momento posterior 3 novas tarefas sejam alocadas para r_k sem que as tarefas anteriores tenham terminado, conclui-se então que r_k passará a executar 7 tarefas ao mesmo tempo. E, portanto, r_k passará a operar com potência p_{k9}. É importante ressaltar que o resultado da razão CO_i/C_i deve ser arredondado para ter apenas uma casa decimal de modo que o nível de carga possa ser determinado. O arredondamento deve ser para cima (nível de carga superior) quando a parte centesimal for maior ou igual a 0,05 e para baixo (nível de carga inferior) quando menor que 0,05.

A seção 6.2.1 explica o funcionamento da estratégia GGreen enquanto a seção 6.2.2 discute os detalhes do algoritmo que a implementa.

6.2.1 Funcionamento de GGreen

Conforme mencionado anteriormente, GGreen propõe uma solução gulosa para resolver o problema da alocação de recursos em grid sob o olhar da computação verde. O objetivo da estratégia é priorizar o recurso que esteja operando com menor potência. Tal decisão promove um uso mais equilibrado dos recursos do grid, entretanto a escolha "gananciosa" pelo recurso com menor potência pode levar a aumento do tempo de execução e, consequentemente, aumentando o consumo de energia.

Inicialmente, a estratégia consiste em processar uma lista de n tarefas (decrescentemente ordenadas pelo peso) que devem ser destinadas a m recursos

disponíveis. Sendo assim, para cada tarefa t_k (onde k = 1, 2, …, n) selecionada pelo escalonador, GGreen encontrará o recurso r_{minPot}.

O recurso r_{minPot} representa aquele cujo valor de potência corresponde ao mínimo dentre todos os m recursos no instante da alocação da tarefa. Ou seja, à medida que uma tarefa é selecionada da lista, as potências dos recursos são analisadas a fim de encontrar o recurso que irá operar com menor potência (em watts) após receber a tarefa. Tal recurso é denominado r_{minPot}. A seção seguinte descreve em detalhes o algoritmo construído para a proposta definida em GGreen.

6.2.2 Algoritmo GGreen

Ao longo desta seção será apresentado o algoritmo capaz de prover uma solução para a estratégia GGreen. O pseudocódigo do algoritmo GGreen pode ser encontrado na Tabela 3 e será explicado a seguir.

Tabela 3: Pseudocódigo do algoritmo GGreen

Algoritmo de Alocação de Tarefas – GGreen

```
1
2   // declarações
3   T = {t1,...,tn}  // lista de tarefas ordenadas decrescentemente pelo peso
4   R = {r1,...,rm}  // lista de recursos do grid disponíveis retornada pelo SI
5   OP[][]   // matriz de entrada m x l com as potências dos recursos
6   C[] // vetor de m posições que representa o total de núcleos de cada recurso
7   CO[] // vetor de m posições que representa a quantidade de núcleos ocupados
8   round(v) // retorna o valor arredondado de v
9
10   // retorna o índice do recurso que irá operar com menor potência após receber a nova tarefa
11  IdMinPotencia()
12     menorInd   // índice do recurso de menor potência
13     menorPot = 100000;
14
15     for each ri in R do
16         l = round(CO[i]+1/C[i]*10);  // obtém o nível de carga do recurso
17         if (OP[i][l] < menorPot)
18             menorPot = OP[i][l];
19             menorInd = i;
20         end if
21     end for
22     return menorInd;   // retorna o índice do recurso com menor potência
23  end IdMinPotencia
24
25   // programa principal
26   for each tk in T do
27     minPot = IdMinPotencia()
28     aloca(tk, rminPot)   // aloca a tarefa tk para o recurso rminPot
29   end for
30
```

Inicialmente, são feitas as declarações de variáveis e funções utilizadas pelo algoritmo (linhas 3-8). O programa principal possui apenas um *loop* (linhas 26-29) que consiste em iterar sobre a lista de tarefas `T`. A lista `T` é previamente ordenada decrescentemente pelo peso, de forma idêntica ao realizado na fase de priorização do algoritmo HGreen. Para cada tarefa t_k da lista, busca-se o índice do recurso com menor potência, através da chamada à função `IdMinPotencia` (linha 27).

A função `IdMinPotencia` busca encontrar o recurso com menor potência quando acrescido de uma tarefa, a fim de retornar seu índice (linhas 11-23). A busca ocorre através da iteração sobre a lista `R`. Para cada recurso, o nível de carga é calculado através da seguinte fórmula: `round(CO[i]+1/C[i]*10)` (linha 16). Caso a potência encontrada neste nível seja menor que a última potência obtida (`menorPot`), as variáveis `menorPot` e `menorInd` são atualizadas (linhas 17-20) e o próximo recurso de `R` passa a ser avaliado. Após o último recurso da lista ser avaliado, a função `IdMinPotencia` é encerrada retornando o conteúdo da variável `menorInd`.

Em seguida, a execução prossegue para o programa principal que realiza a alocação da tarefa t_k no recurso r_{minPot} (linha 28). O processo então se repete para a próxima tarefa da lista e segue até que todas as tarefas de `T` tenham sido processadas.

6.3 Estratégia Multiobjetivo

Nos últimos anos, a otimização multiobjetivo (ou multicritério) tem se tornado uma ferramenta de longo alcance devido em parte ao aumento do poder computacional disponível. O conceito de otimização multiobjetivo pode ser definido como o processo de simultaneamente otimizar dois ou mais objetivos conflitantes sujeitos a certas restrições (STEUER, 1986). Ou seja, não existe um único ótimo global, assim como ocorre na busca do máximo ou mínimo de uma função. Em vez disso, existe um conjunto de ótimos que satisfazem, de formas distintas, os diferentes objetivos envolvidos na análise.

Diversos problemas reais de diferentes áreas envolvem a otimização de funções objetivo distintas. Alguns exemplos são:

- *Maximizar lucro* e *minimizar o custo* de um produto.
- Considerando a busca de um emprego, *maximizar salário* e *minimizar distância do local de trabalho*.

- *Minimizar taxa de acidentes de trabalho* e *minimizar custos*.
- *Maximizar o desempenho* e *minimizar o consumo de combustível* de um carro.

Em problemas multiobjetivo, conforme os citados anteriormente, não se pode identificar uma solução única que simultaneamente otimize cada objetivo. A busca por soluções ótimas leva a um ponto em que quando se tenta melhorar um objetivo, penaliza os outros objetivos como resultado da nova solução. Neste caso, as soluções "piores" são ditas dominadas por soluções "melhores" e podem ser descartadas. Por exemplo, considerando um problema para maximizar três objetivos, a solução (5,3,4) é dita dominar a solução (4,3,4), pois a primeira solução melhora o primeiro objetivo sem reduzir os outros. Sendo assim, uma determinada solução é chamada de não-dominada ou Pareto-ótimo se a mesma não pode ser eliminada do conjunto solução final, sendo substituída por outra que melhore um objetivo sem penalizar os demais objetivos. Encontrar soluções não-dominadas e quantificar os compromissos (*trade-offs*) negociando entre diferentes objetivos é a principal tarefa quando se está modelando e resolvendo um problema de otimização multiobjetivo.

Considerando o problema de alocação de recursos em grid desenvolvido no contexto deste trabalho, dois objetivos relevantes são observados: a *minimização do tempo de execução* (tempo de resposta) e a *minimização do consumo de energia*. A análise minuciosa de resultados de *benchmarks* em diferentes plataformas computacionais permite a conclusão de que tais objetivos são conflitantes em diversas ocasiões. Ou seja, existem recursos que possuem elevado grau de desempenho, proporcionando baixo tempo de resposta para a execução de uma dada tarefa, porém consomem mais energia quando comparados a outros recursos de desempenho inferior. Tal fato também acontece no sentido inverso, ou seja, um recurso muito econômico energeticamente que demanda elevado tempo de execução.

Mediante tal constatação, o problema da alocação de recursos em grid conforme tratado neste trabalho pode ser estudado como um problema de otimização multiobjetivo. As seções seguintes discorrem sobre o tema, apresentando inicialmente um modelo matemático do problema, seguido pela discussão do algoritmo proposto como solução e da análise do algoritmo frente a uma instância do problema.

6.3.1 Formulação do Problema

Antes de descrever o problema formalmente, é preciso fazer algumas considerações. Cada máquina representa um recurso de execução do grid e contém múltiplos núcleos, o que a permite executar mais de uma tarefa ao mesmo tempo. Além disso, cada tarefa executa inteiramente em um único núcleo e o número de tarefas designadas para uma máquina não pode ultrapassar a sua quantidade de núcleos. As dependências entre tarefas não são consideradas.

Considere X o conjunto de todas as alocações possíveis de um lote de n tarefas a m máquinas disponíveis do grid. Seja x_{ij} a variável de decisão (cujos índices i=1,2,…,n e j=1,2,…,m indicam tarefa e máquina, respectivamente), onde $x_{ij} = 1$ significa que a tarefa T_i será executada na máquina M_j e $x_{ij} = 0$ significa o oposto (a não existência da alocação). A seguir, é apresentado o modelo matemático que representa a formulação do problema:

$$\text{minimizar} \quad \max \left\{ t_{ij} \mid x_{ij} = 1 \right\}$$

$$\text{minimizar} \quad \sum_{i=1}^{n} \sum_{j=1}^{m} e_{ij} x_{ij}$$

$$\text{sujeito a :} \quad \sum_{j=1}^{m} x_{ij} = 1, \quad i = 1, \ldots, n$$

$$\sum_{i=1}^{n} x_{ij} \leq C_j, \quad j = 1, \ldots, m$$

$$x_{ij} \in \{0,1\}, \quad i = 1, \ldots, n \text{ e } j = 1, \ldots, m.$$

Onde t_{ij} e e_{ij} são dados de entrada que representam o tempo de execução e o consumo de energia da tarefa T_i quando executada na máquina M_j, respectivamente. Tais valores podem ser obtidos por meio de estimativas a partir de execuções anteriores e do uso métricas de desempenho e consumo de energia fornecidas por *benchmarks*, conforme explicado anteriormente.

Observe ainda que o primeiro grupo de restrições garante que toda tarefa será executada em uma única máquina, enquanto o segundo grupo garante que o total de tarefas destinadas a uma máquina não ultrapassa a quantidade de núcleos livres na mesma, valor definido por C. Esta última restrição busca evitar a formação de filas nos escalonadores locais, em consequência de uma possível concentração de alocações de recursos com boas métricas de desempenho e consumo de energia.

6.3.2 Algoritmo BOTEEN

Nesta seção é discutida uma solução algorítmica para a obtenção de um conjunto mínimo completo de alocações Pareto-ótimas considerando o problema bi-critério (tempo e energia) modelado na seção anterior. A solução foi alcançada através do algoritmo denominado BOTEEN (BiObjetivo TEmpo ENergia), o qual é apresentado na Tabela 4. BOTEEN teve seu desenvolvimento baseado em soluções tratadas em conhecidos trabalhos de otimização multiobjetivo (BERMAN *et al.*, 1990) (BORNSTEIN *et al.*, 2012).

Inicialmente, são declaradas as variáveis e funções utilizadas no algoritmo (linhas 3-11). O programa principal tem início a partir da chamada à função `MinEnergia` (linha 35), a qual recebe como parâmetro um limite superior para o tempo de execução, simbolizado por ∞. A função `MinEnergia` (linhas 15-32) retorna uma alocação completa para as *n* tarefas da lista *T* com custo energético mínimo e com tempo de execução máximo menor que o valor recebido como parâmetro: `trade-off`. `MinEnergia` pode não ser capaz de retornar uma alocação completa pelo fato de não atender ao compromisso do tempo máximo ser menor que valor do `trade-off`. Neste caso, `Aloc` ← ∅ e nenhuma solução é retornada.

Na primeira chamada a `MinEnergia`, o valor de `trade-off` é o limite superior de todos os tempos de execução (∞). Sendo assim, a função retorna para $\texttt{Aloc}_1$ (linha 35) uma alocação com custo energético mínimo independente dos tempos de execução. Em seguida, inicia-se um *loop* (linhas 36-46) que começa buscando uma nova alocação $\texttt{Aloc}_2$ através de `MinEnergia` com `trade-off` igual ao tempo de execução máximo de $\texttt{Aloc}_1$ (linha 37). Ou seja, $\texttt{Aloc}_2$ poderá assumir dois valores: (a) valor vazio (∅); ou (b) uma alocação completa das *n* tarefas com tempo máximo menor que o tempo máximo de $\texttt{Aloc}_1$.

No caso do item (a), $\texttt{Aloc}_1$ é inserida na lista `Sol` (conjunto solução) e o *loop* é interrompido (linhas 38-40). Em se tratando do item (b), caso o total de consumo de energia de $\texttt{Aloc}_2$ seja maior ou igual ao consumo de $\texttt{Aloc}_1$, $\texttt{Aloc}_1$ será inserida na lista `Sol` (linhas 42-43). Caso contrário, $\texttt{Aloc}_1$ recebe o conteúdo de $\texttt{Aloc}_2$ (linha 44) e segue-se para a próxima iteração do *loop*.

Tabela 4: Pseudocódigo do algoritmo BOTEEN

	Algoritmo BOTEEN
1	
2	// declarações
3	$T = \{T_1, ..., T_n\}$ // lista de tarefas ordenadas decrescentemente pela carga
4	$M = \{M_1, ..., M_m\}$ // lista de máquinas ordenadas decrescentemente por eficiência energética
5	TE[][] // matriz de entrada *n* x *m* com os tempos de execução
6	EN[][] // matriz de entrada *n* x *m* com os consumos de energia
7	$Aloc_1$[] // vetor de *n* posições que representa uma alocação para as tarefas da lista *T*
8	$Aloc_2$[] // vetor de *n* posições que representa uma alocação para as tarefas da lista *T*
9	$Sol = \{Aloc_1, ..., Aloc_k\}$ // lista de alocações Pareto-ótimas (conjunto-solução)
10	tMax(Aloc[]) // função que retorna o tempo de execução máximo da alocação *Aloc[]*
11	eTotal(Aloc[]) // função que retorna o total do consumo de energia da alocação *Aloc[]*
12	
13	// busca uma alocação das *n* tarefas da lista *T* com custo energético mínimo e com tempo
14	// máximo menor que *trade-off*
15	MinEnergia(trade_off)
16	// declarações
17	C[] // vetor de *m* posições que representa a qtde de núcleos em cada máquina
18	QtdeT[] // vetor de *m* posições que representa a qtde de tarefas alocadas em cada máq.
19	Aloc[] // vetor de *n* posições que representa uma alocação para as tarefas da lista *T*
20	
21	QtdeT[] ← {0,...,0} // inicialmente, todas as máquinas estão vazias (sem tarefas)
22	i ← 1; j ← 1;
23	while (i ≤ n) && (j ≤ m)
24	if (QtdeT[j] < C[j]) && (TE[i][j] < trade_off)
25	Aloc[i] ← j; QtdeT[j]++; i++; j ← 1;
26	else
27	j++;
28	end while
29	if (j = m+1)
30	Aloc ← ∅;
31	return Aloc;
32	end MinEnergia
33	
34	// programa principal
35	$Aloc_1 \leftarrow$ MinEnergia(∞)
36	while (true) do
37	$Aloc_2 \leftarrow$ MinEnergia(tMax($Aloc_1$))
38	if ($Aloc_2$ = ∅)
39	insere(Sol, $Aloc_1$);
40	break;
41	else
42	if (eTotal($Aloc_1$) < eTotal($Aloc_2$))
43	insere(Sol, $Aloc_1$);
44	$Aloc_1 \leftarrow Aloc_2$;
45	end if
46	end while
47	

6.3.3 Instância do Problema

Para efeitos didáticos esta seção elabora uma pequena instância do problema discutido na seção 6.3.1. Considere um ambiente de grid com 3 máquinas onde se deseja executar 4 tarefas. Algumas notações são necessárias para o entendimento do

exemplo: Op_i representa o tamanho (peso) da tarefa T_i em operações ponto flutuante (opf); S_j indica a quantidade de opf que um núcleo de processamento da máquina M_j executa por segundo; W_j representa a quantidade de opf executada por um núcleo de processamento da máquina M_j para cada watt consumido pela mesma; e finalmente, C_j é a quantidade de núcleos de processamento disponíveis na máquina M_j.

Além disso, as medidas t_{ij} e e_{ij} correspondem ao tempo em segundos e energia em watts gastos para executar a tarefa T_i na máquina M_j, respectivamente. Essas medidas são calculadas da seguinte forma: $t_{ij} = Op_i / S_j$ e $e_{ij} = Op_i / W_j$.

O problema exemplificado nesta seção é retratado pela Figura 5, a qual apresenta um grafo bipartido completo onde os vértices mais à esquerda representam as tarefas e aqueles à direita representam as máquinas do grid. Cada aresta do grafo indica a possível designação de uma tarefa T_i para ser executada em uma máquina M_j. Os pesos t_{ij} e e_{ij} são atribuídos a cada uma das arestas.

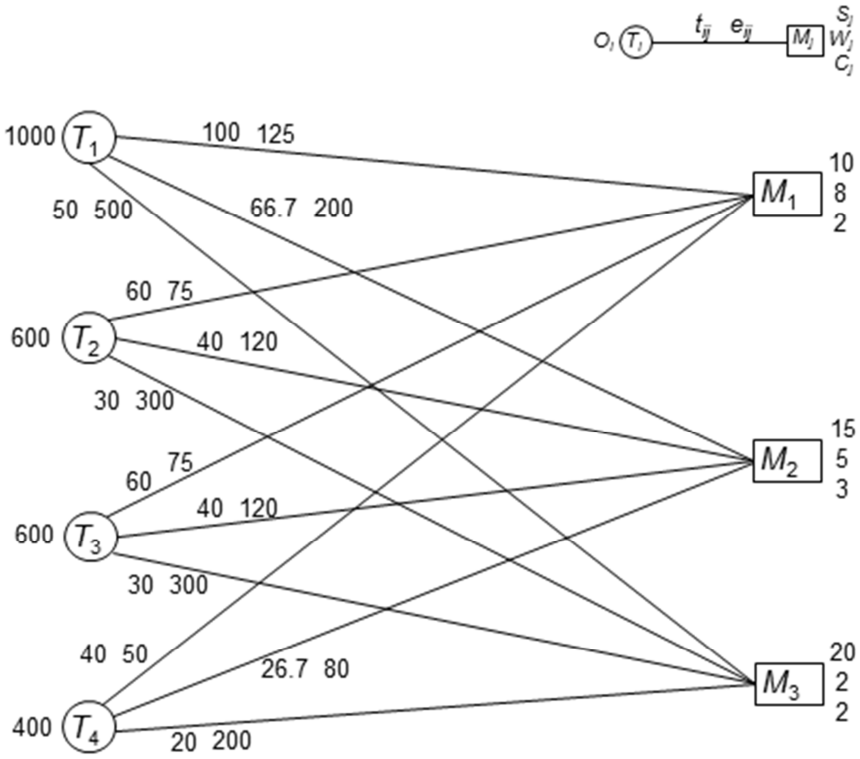

Figura 5: Instância do problema representada como um grafo

A Figura 6 exibe o *conjunto mínimo completo de soluções Pareto-ótimas* para a instância representada pela Figura 5. Cada solução corresponde à alocação de todas as tarefas (T_1, T_2, T_3 e T_4) para alguma(s) máquina(s) (M_1, M_2 e M_3) respeitando a disponibilidade C_j.

O conjunto de soluções contém quatro alocações Pareto-ótimas denominadas A, B, C e D, seguindo a sequência retornada pelo algoritmo BOTEEN. Essas alocações podem ser identificadas pelas arestas tracejadas na Figura 6. Inicialmente, percebe-se que todas as arestas são consideradas na solução A (o *trade-off* é ∞). Posteriormente, à medida que o trade-off vai diminuindo, o grafo vai sendo podado com algumas arestas sendo descartadas.

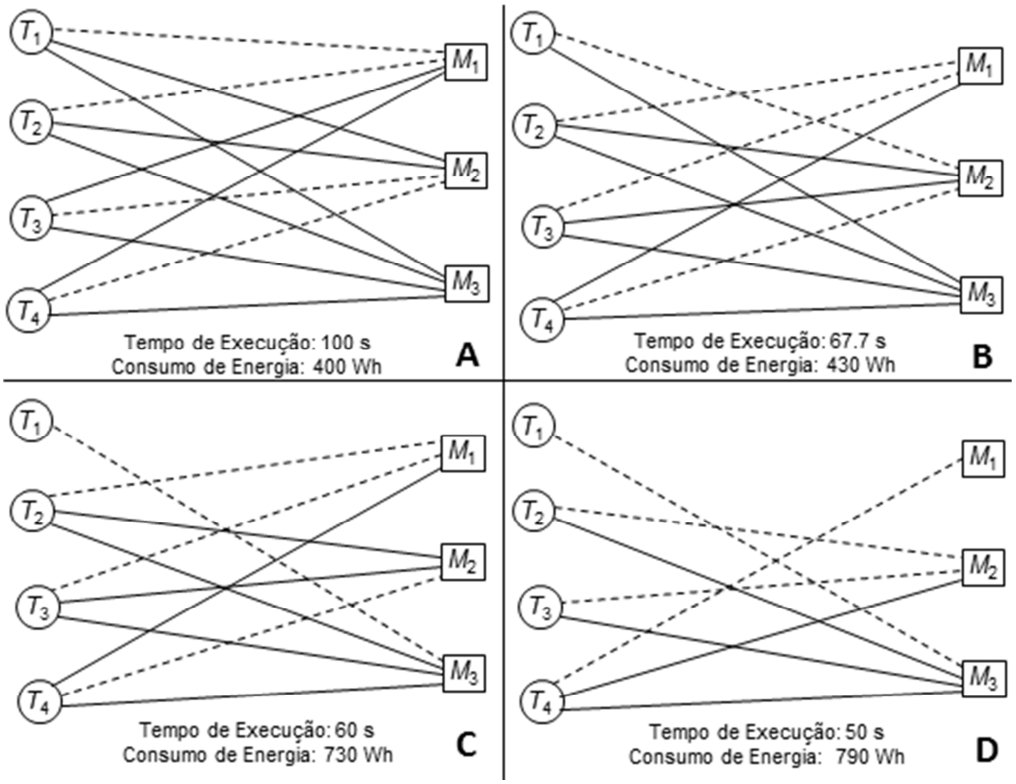

Figura 6: Alocações Pareto-ótimas retornadas por BOTEEN

Como o algoritmo BOTEEN retorna um conjunto de soluções (a lista `Sol`), pode ser interessante utilizar um método para selecionar da lista a alocação final a ser aplicada pelo escalonador global (meta-escalonador). Tal método poderia priorizar um dos dois objetivos ou mesmo buscar a alocação mais próxima da mediana para ambos os objetivos.

Capítulo 7 – Análise das Estratégias Propostas

Neste capítulo são apresentados os resultados obtidos a partir da análise das estratégias discutidas no Capítulo 6. A abordagem utilizada para avaliar as estratégias de alocação divide-se em duas partes. Na primeira foi construído um ambiente de simulação do grid sobre o qual foram desenvolvidas soluções em linguagem Java para as estratégias HGreen e GGreen. Na segunda parte, a estratégia BOTEEN foi avaliada através de um programa desenvolvido em linguagem C que permitiu gerar as alocações Pareto-ótimas para diferentes cenários a partir de dados de entrada previamente estimados.

Sendo assim, este capítulo encontra-se organizado de maneira que a seção 7.1 discute os detalhes de implementação e apresenta os resultados da simulação para HGreen e GGreen, enquanto a seção 7.2 analisa os resultados obtidos para a estratégia BOTEEN.

7.1 Resultados da Simulação: HGreen e GGreen

Grids computacionais possuem características dinâmicas e heterogêneas. Nos moldes considerados neste trabalho, o ambiente de grid possui ainda dimensões e infraestrutura de grande magnitude. Em virtude disso, torna-se difícil ou quase impossível avaliar o desempenho de um escalonador de grid de forma controlada e reproduzível para diferentes cenários em um ambiente real.

Uma alternativa interessante para contornar este problema é utilizar ferramentas que permitam a simulação de um ambiente de grid. Atualmente, existem diversos sistemas simuladores de grid: OptorSim (NAQVI, RIGUIDEL, 2005), GrenchMark (IOSUP, EPEMA, 2006), GridNet (LAMEHAMEDI *et al.*, 2003), GridSim (BUYYA, MURSHED, 2002), JFreeSim (JIN *et al.*, 2005), NSGrid (VOLCKAERT *et al.*, 2003) e SimGrid (CASANOVA, 2001).

Os trabalhos supracitados são alguns exemplos dos mais conhecidos sistemas para simulação de grids. Esses simuladores possuem características e abordagens muitas vezes distintas. Portanto, a decisão sobre qual sistema utilizar envolve diferentes aspectos tais como: arquitetura e tecnologia aplicada; processo de simulação; modelagem da rede; modelagem da carga de trabalho e modelagem dos recursos computacionais.

Em se tratando do problema explorado nesta tese, um aspecto importante seria o modelo de energia. Ou seja, se o simulador de grid seria capaz de capturar o consumo energético dos recursos modelados. Dentre os modelos avaliados durante a fase inicial da implementação foi constatado que nenhum deles atendia de forma nativa a tal exigência. Diante desse fato, a decisão sobre qual simulador utilizar neste trabalho foi orientada pelos seguintes critérios: arquitetura extensível; código fonte aberto; conhecimento da linguagem empregada; facilidade de uso e especificação de recursos heterogêneos com múltiplos núcleos.

Levando esses pontos em consideração, além do fato de ser uma ferramenta bastante utilizada pela comunidade científica, o sistema GridSim (BUYYA, MURSHED, 2002) foi escolhido para simular o ambiente de grid que serviria de testes para a aplicação das estratégias HGreen e GGreen. A seção seguinte discorre sobre as características do simulador e como o mesmo foi utilizado para medir o consumo de energia dos recursos do grid.

7.1.1 GridSim: Conceitos Básicos

GridSim é construído sobre a infraestrutura de um simulador dirigido a eventos discretos para ambientes Java, conhecido como SimJava (HOWELL, MCNAB, 1998). SimJava é uma proposta geral que contém uma série de entidades que executam em paralelo cada uma em sua própria linha de execução (*multithreading*).

GridSim utiliza o conceito de PEs (Elementos de Processamento) para modelar recursos computacionais, onde um ou mais PEs podem ser agrupados para modelar uma máquina. E uma ou mais máquinas podem também ser combinadas para criar um grid ou cluster. O recurso possui ainda uma velocidade de processamento que é expressa em MIPS (Milhões de Instruções Por Segundo). As características de cada recurso são registradas junto ao serviço de informação do grid (GIS). GIS é responsável por manter uma lista dos recursos disponíveis do grid.

O modelo de aplicação de GridSim baseia-se no conceito de objetos chamados de *Gridlets*. Um *Gridlet* representa um pacote que possui todas as informações referentes à tarefa a ser executada, incluindo o tamanho da tarefa expresso em MI (Milhões de Instruções).

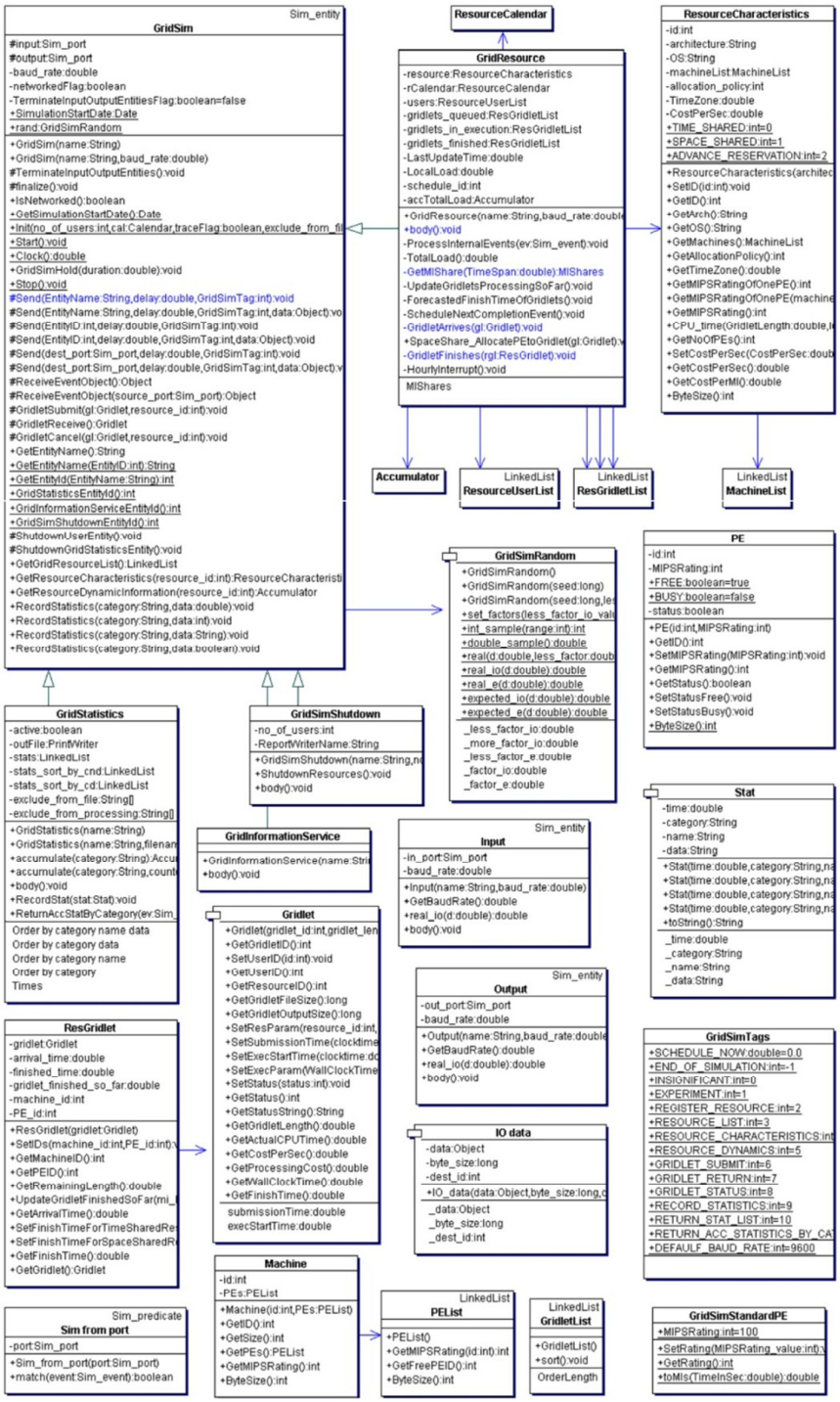

Figura 7: Diagrama de Classes do simulador GridSim

Outro componente importante do simulador GridSim é o *Broker*, o qual é responsável pelo escalonamento de tarefas no grid. O processo de escalonamento funciona da seguinte maneira. Inicialmente, o usuário do grid dispõe de um conjunto de tarefas (*gridlets*) para ser executado. Essas tarefas são submetidas ao *Broker* que por sua

vez consulta os recursos disponíveis via GIS. Em seguida, a política de escalonamento definida pelo *Broker* é aplicada para decidir como as tarefas serão alocadas.

Detalhes sobre o modelo de representação dos componentes de rede não são comentados por não serem de interesse deste trabalho. A Figura 7 apresenta uma visão completa do diagrama de classes do simulador GridSim.

7.1.2 Construção do Ambiente de Simulação

Nesta seção são apresentadas algumas considerações acerca da simulação do ambiente de grid implementada para avaliar a execução das estratégias HGreen e GGreen.

Conforme mencionado anteriormente GridSim e os demais simuladores ainda não possuem o consumo de energia como parte de seu modelo de recursos. Sendo assim, foi preciso adaptar o simulador para que a medição do consumo energético pudesse ser realizada durante a simulação.

A alternativa encontrada para superar esse obstáculo foi utilizar o conceito de custo presente no modelo de recursos para atender ás necessidades de grids utilitários. O custo representa a quantia a ser paga pelo uso do recurso. O valor estipulado para o custo representa a quantia a ser cobrada do usuário para cada segundo que o recurso estiver sendo usado pelo mesmo.

A estratégia para adaptar o custo pelo uso em consumo energético consiste em atribuir o valor da potência em que estará operando o recurso ao atributo *CostPerSec*. O valor da potência é obtido a partir do *benchmark* SPECpower de acordo com o nível de carga da máquina no momento da execução, calculado de forma idêntica ao que foi explicado na seção 6.2.

Sendo assim, ao final da execução de um lote de tarefas é possível calcular o consumo de energia de um recurso através do seu valor de custo (de acordo com a variação de sua carga) multiplicado pelo tempo de execução. O valor do consumo energético assim obtido é expresso em watts-segundos e pode ser facilmente convertido para a medida mais conhecida: quilowatt-hora (kWh).

Para proceder a representação dos recursos do grid no simulador GridSim as classes *GridResource* e *GridInformationService* foram estendidas pelas classes *GreenGridResource* e *GreenGridInformationService*, respectivamente. À classe *GreenGridResource* foram adicionados os atributos que armazenam as potências para os 11 diferentes níveis de carga. Além disso, foi criada a classe *GreenTags* para permitir

o registro dessas potências junto ao serviço de informação do grid (classe *GreenGridInformationService*).

Os algoritmos HGreen e GGreen foram implementados a partir das classes *HGreenCaseStudies* e *GGreenCaseStudies*, respectivamente. Sendo a primeira, responsável pela criação das tarefas, conforme será explicado nas próximas seções. As principais classes adicionadas ao simulador podem ser vistas em destaque na Figura 8.

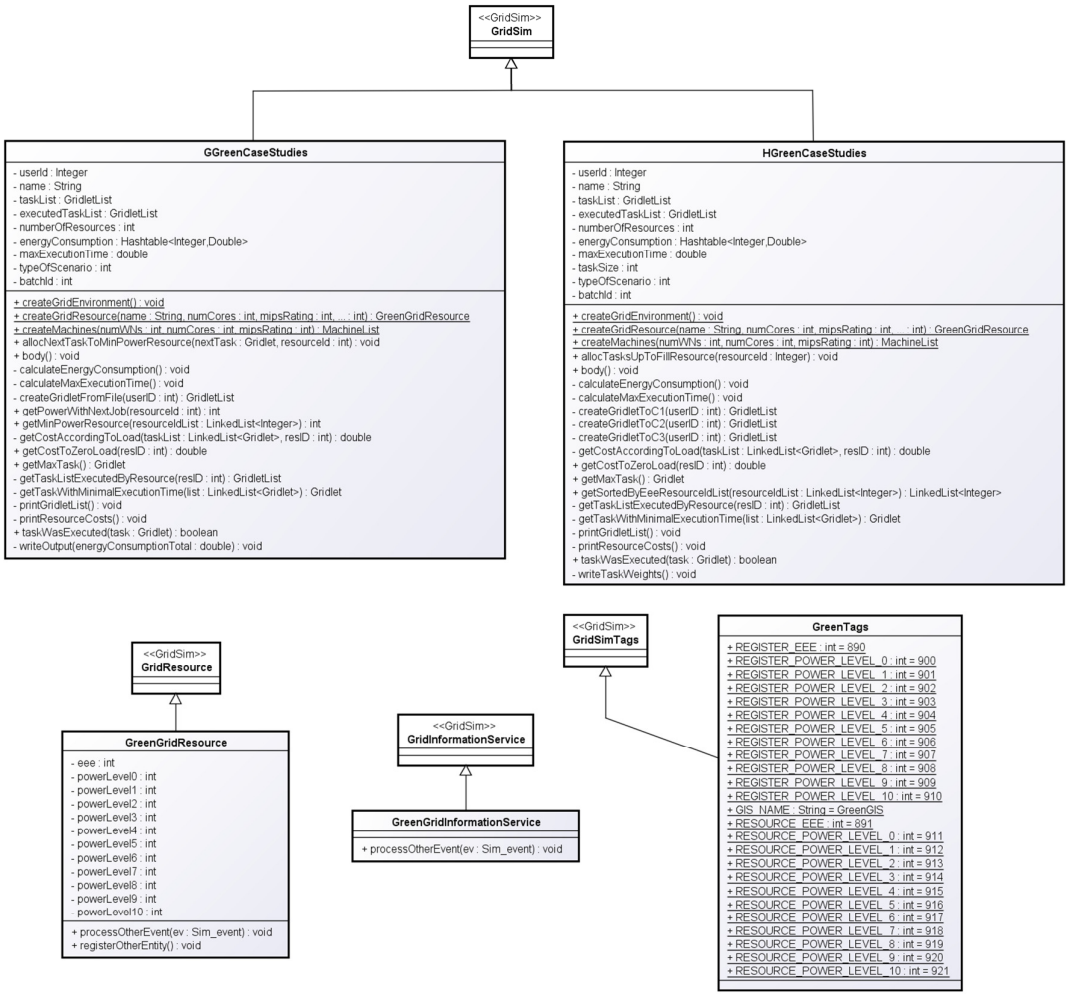

Figura 8: Classes Estendidas de GridSim para implementar HGreen e GGreen

Os códigos fontes das principais classes podem ser encontrados nos Apêndices 2 e 3. As seções a seguir descrevem os cenários criados para avaliar as estratégias HGreen e GGreen.

7.1.3 Descrição dos Recursos do Grid

Esta seção descreve o ambiente de grid construído a partir do simulador explicado na seção anterior. Para todos os cenários criados para analisar as estratégias

HGreen e GGreen foi utilizado um único grid dispondo de 6 sítios: S_1, S_2, S_3, S_4, S_5 e S_6. E, por questões de simplicidade, cada sítio dispõe de um único recurso de execução para o qual a tarefa poderá ser destinada pelo escalonador.

A Tabela 5 contém as principais informações referentes à configuração dos recursos do grid. As métricas de desempenho (MIPS) e eficiência energética (eee) foram obtidas partir do *benchmark* HEPSPEC e SPECpower, respectivamente.

Tabela 5: Configuração dos recursos de execução do Grid

Sítio	Configuração	Cores	MIPS	eee
S_1	Supermicro 6025B (Intel Xeon E5345)	8	1174	435
S_2	HP ProLiant DL380 (Intel Xeon 5160)	4	1535	382
S_3	HP ProLiant DL360 (Intel Xeon L5530)	8	2548	1586
S_4	Dell 2970 (AMD Opteron 2356)	8	1190	545
S_5	Bull SAS R440 (Intel Xeon X5670)	12	2902	2831
S_6	SGI Altix XE320 (Intel Xeon E5420)	8	1322	681

Para cada um dos recursos foram obtidos, também a partir do *benchmark* SPECpower, os valores de suas potências considerando os 11 diferentes níveis de carga, conforme explicado no capítulo 6. Sendo assim, a Tabela 6 exibe em formato de matriz as potências atingidas pelos recursos apresentados na Tabela 5 na medida em que variam sua carga de trabalho ($l = 0, 1, \ldots, 9, 10$).

Tabela 6: Potências dos recursos do grid de acordo com a carga de trabalho

	S_1	S_2	S_3	S_4	S_5	S_6
$l = 0$	219	172	88	139	60	155
$l = 1$	234	177	117	164	107	172
$l = 2$	248	182	131	185	122	185
$l = 3$	260	187	144	205	133	197
$l = 4$	273	195	155	223	144	206
$l = 5$	286	205	167	236	157	214
$l = 6$	298	218	183	249	173	221
$l = 7$	309	229	197	263	191	231
$l = 8$	318	242	212	277	211	247
$l = 9$	327	252	230	292	229	255
$l = 10$	334	258	245	302	247	260

7.1.4 Descrição das Tarefas

Esta seção descreve o conjunto de tarefas utilizado para a análise das estratégias HGreen e GGreen. Com o intuito de identificar possíveis tendências ou variações de comportamento das estratégias propostas, foram definidos 3 tipos de cenários que expressam realidades distintas. O primeiro cenário C1 possui uma distribuição com alta variância da carga das tarefas (peso das tarefas definidos em MI). O cenário C2 possui uma distribuição mais equilibrada (baixa variância). Finalmente, o cenário C3 apresenta taxa de variância nula, ou seja, todas as tarefas possuem o mesmo peso (são idênticas).

Para cada um dos cenários citados anteriormente, foram gerados 50 lotes de tarefas distintos (LT101 a LT150) para os quais HGreen e GGreen foram executadas. Apesar de serem distintos entre si, os lotes pertencentes a cada cenário mantêm as propriedades inerentes ao cenário, ou seja, variância alta, baixa ou nula. Sendo assim, tem-se um total de 150 lotes de tarefas distribuídos entre os cenários C1, C2 e C3. Dentro de cada cenário os pesos das tarefas foram obtidos aleatoriamente a partir de uma faixa de valores que garante a propriedade do cenário.

Considerando que o grid descrito na seção anterior dispõe de um total de 48 núcleos, as estratégias HGreen e GGreen foram avaliadas a partir de séries com subconjuntos de 12, 24, 36 e 48 tarefas, expressando possíveis momentos distintos da carga do grid. Desta forma, cada lote destacado anteriormente resultará em 4 lotes, uma vez que o executaremos em 4 subconjuntos. Ou seja, em vez de um total de 150 lotes de tarefas o teste consistiu na execução de um total de 600 lotes de tarefas. A título de exemplificação, as Tabelas 7 e 8 apresentam os pesos de um lote de 48 tarefas para os cenários C1 e C2, respectivamente.

Tabela 7: Lote com 48 tarefas do cenário C1 (alta variância)

C1_LT137							
Id	**MI**	**Id**	**MI**	**Id**	**MI**	**Id**	**MI**
1	*160000*	13	*150000*	25	*150000000*	37	*430000000*
2	*150000*	14	*170000*	26	*310000000*	38	*460000000*
3	*160000*	15	*410000*	27	*180000000*	39	*450000000*
4	*330000*	16	*340000*	28	*190000000*	40	*170000000*
5	*210000*	17	*370000*	29	*410000000*	41	*150000000*
6	*420000*	18	*270000*	30	*480000000*	42	*240000000*
7	*400000*	19	*420000*	31	*190000000*	43	*200000000*
8	*330000*	20	*450000*	32	*160000000*	44	*310000000*
9	*380000*	21	*220000*	33	*430000000*	45	*170000000*
10	*380000*	22	*400000*	34	*190000000*	46	*210000000*
11	*400000*	23	*430000*	35	*440000000*	47	*400000000*
12	*440000*	24	*200000*	36	*270000000*	48	*260000000*

Conforme evidenciam os pesos indicados nas tabelas, o lote LT137 de C1 reúne tarefas muito leves e muito pesadas enquanto o lote LT137 de C2 possui uma distribuição mais uniforme. Finalmente, os lotes do cenário C3 apresentam taxa de variância nula, ou seja, todas as suas tarefas possuem o mesmo peso (são idênticas). O valor do peso único definido para cada lote de C3 é obtido aleatoriamente dentro do seguinte intervalo: *5000000* a *500000000* MI. Por questões de espaço, não são apresentados os pesos dos 600 lotes submetidos durante os testes das estratégias.

Tabela 8: Lote com tarefas do cenário C2 (baixa variância)

C2_ LT137							
Id	**MI**	**Id**	**MI**	**Id**	**MI**	**Id**	**MI**
1	*2500000*	13	*3800000*	25	*21000000*	37	*250000000*
2	*2400000*	14	*1300000*	26	*10000000*	38	*410000000*
3	*1600000*	15	*900000*	27	*42000000*	39	*180000000*
4	*1000000*	16	*4400000*	28	*19000000*	40	*290000000*
5	*4600000*	17	*32000000*	29	*47000000*	41	*490000000*
6	*800000*	18	*36000000*	30	*13000000*	42	*350000000*
7	*4000000*	19	*49000000*	31	*31000000*	43	*320000000*
8	*1400000*	20	*13000000*	32	*30000000*	44	*320000000*
9	*4600000*	21	*32000000*	33	*190000000*	45	*70000000*
10	*3100000*	22	*40000000*	34	*140000000*	46	*210000000*
11	*1400000*	23	*29000000*	35	*50000000*	47	*220000000*
12	*4000000*	24	*40000000*	36	*150000000*	48	*150000000*

No estado atual deste trabalho, não é considerada a existência de dependências entre as tarefas. Ou seja, os lotes mencionados anteriormente possuem tarefas independentes. Outra simplificação diz respeito a não consideração do volume da entrada e saída de dados das tarefas. As seções seguintes discutem os resultados encontrados por HGreen e GGreen.

7.1.5 Resultados de HGreen e GGreen

Antes de apresentar os resultados, é importante destacar que para todos os cenários e séries executadas em HGreen, foi considerado $fv = 1$. Ou seja, a economia de energia é colocada como meta principal. Além disso, considera-se $tde = 0$ para todas as tarefas, uma vez que a entrada/saída de dados é descartada nesta avaliação. Desta forma, pode-se dizer que a eficiência de execução é dada por $eee(r_i)$ independente da tarefa a ser executada.

A Tabela 9 apresenta os consumos de energia retornados pela simulação de HGreen e GGreen considerando os lotes gerados para o cenário C1.

Tabela 9: Resultados de HGreen e GGreen em kWh para os lotes de C1

Lote	12 Tarefas		24 Tarefas		36 Tarefas		48 Tarefas	
	HGreen	**GGreen**	**HGreen**	**GGreen**	**HGreen**	**GGreen**	**HGreen**	**GGreen**
LT101	435	458	452	565	494	725	553	947
LT102	406	463	445	503	464	706	553	1008
LT103	380	391	473	611	515	862	493	622
LT104	381	422	421	486	514	940	555	1048
LT105	310	350	440	521	513	841	560	913
LT106	442	512	402	493	504	768	527	836
LT107	416	429	444	473	476	611	552	930
LT108	381	443	463	609	499	810	589	1048
LT109	398	415	462	527	491	830	526	903
LT110	426	466	464	519	501	662	560	1000
LT111	428	488	444	625	478	724	569	1069
LT112	398	433	398	578	474	741	528	848
LT113	304	344	465	540	501	763	536	858
LT114	389	400	475	755	473	750	607	1003
LT115	425	464	445	586	506	770	568	1064
LT116	338	391	417	453	501	763	518	763
LT117	370	383	473	630	472	658	504	786
LT118	396	406	461	545	504	726	554	908
LT119	366	374	426	642	500	793	504	826
LT120	441	492	447	668	469	664	523	851
LT121	431	442	403	473	474	700	541	754
LT122	399	411	458	664	462	616	536	954
LT123	404	415	429	516	493	686	552	888
LT124	365	422	446	489	494	794	569	1052
LT125	434	485	456	536	502	795	598	995
LT126	404	415	465	518	497	810	560	1015
LT127	367	376	434	582	500	853	528	757
LT128	441	483	465	501	503	765	532	920
LT129	427	476	468	526	498	770	555	1031
LT130	423	434	450	482	468	642	564	1026
LT131	434	505	467	541	460	634	558	1011
LT132	424	437	453	477	502	725	536	862
LT133	364	385	442	540	506	960	543	912
LT134	440	452	468	564	512	838	545	879
LT135	416	472	435	474	513	780	556	907
LT136	417	431	473	536	475	810	566	1068
LT137	422	433	450	481	448	697	514	779
LT138	363	412	436	496	511	815	550	904
LT139	443	514	454	545	509	774	544	988
LT140	400	455	396	514	474	722	534	893
LT141	441	453	390	598	505	748	547	960
LT142	429	480	452	523	471	726	530	860
LT143	441	453	463	498	513	910	537	927
LT144	408	428	443	472	487	656	542	866
LT145	373	423	465	752	502	787	534	865
LT146	413	424	465	548	459	652	546	918
LT147	241	261	381	446	503	705	536	910
LT148	441	454	464	541	515	881	540	959
LT149	388	400	474	652	502	754	550	885
LT150	414	425	439	599	495	875	526	703

Similarmente, as Tabelas 10 e 11 apresentam os resultados de HGreen e GGreen para os lotes dos cenários C2 e C3, respectivamente.

Tabela 10: Resultados de HGreen e GGreen para os lotes de C2

Lote	12 Tarefas		24 Tarefas		36 Tarefas		48 Tarefas	
	HGreen	**GGreen**	**HGreen**	**GGreen**	**HGreen**	**GGreen**	**HGreen**	**GGreen**
LT101	437	506	407	445	450	498	494	567
LT102	365	384	308	318	473	496	487	559
LT103	336	339	452	468	436	493	501	739
LT104	434	442	284	330	477	542	472	522
LT105	333	342	452	466	462	517	493	545
LT106	413	468	397	442	469	551	494	870
LT107	340	347	362	372	368	403	476	546
LT108	409	416	441	494	442	477	462	541
LT109	245	265	441	464	390	407	408	453
LT110	312	316	419	430	466	537	488	538
LT111	400	409	411	421	451	469	500	676
LT112	421	432	313	345	386	431	476	553
LT113	367	375	432	482	433	498	486	630
LT114	237	246	451	524	476	532	484	597
LT115	226	230	425	434	405	454	491	757
LT116	313	320	312	322	473	496	461	528
LT117	386	445	417	425	442	506	452	499
LT118	438	487	439	509	438	476	494	679
LT119	373	380	433	484	416	447	506	589
LT120	374	382	384	394	460	495	494	741
LT121	324	374	456	481	473	537	450	584
LT122	280	286	391	425	449	611	483	823
LT123	394	404	452	506	420	440	444	480
LT124	372	376	429	444	463	484	442	497
LT125	345	348	422	435	452	542	487	540
LT126	326	359	392	404	457	514	507	804
LT127	281	315	379	390	459	570	456	523
LT128	422	478	298	308	447	515	483	564
LT129	213	246	452	496	425	487	483	608
LT130	377	398	451	523	471	523	502	618
LT131	372	376	446	489	466	486	504	869
LT132	394	403	454	468	471	513	464	533
LT133	428	436	308	318	433	487	478	500
LT134	351	360	375	415	456	477	494	699
LT135	422	460	423	436	415	431	488	754
LT136	440	508	371	404	463	485	472	694
LT137	299	308	441	454	357	416	478	500
LT138	360	398	415	435	448	467	479	549
LT139	337	341	412	477	470	578	490	513
LT140	211	216	422	479	383	399	434	517
LT141	376	434	450	476	393	407	474	525
LT142	391	398	456	470	471	517	492	574
LT143	384	393	436	448	417	435	487	542
LT144	438	469	438	452	441	467	499	636
LT145	415	420	446	520	457	536	479	632
LT146	379	382	434	477	463	527	466	545
LT147	438	469	380	430	463	542	426	497
LT148	378	427	383	436	479	697	456	476
LT149	436	446	395	429	469	547	477	615
LT150	401	411	409	466	439	506	490	563

Tabela 11: Resultados de HGreen e GGreen em kWh para os lotes de C3

Lote	12 Tarefas		24 Tarefas		36 Tarefas		48 Tarefas	
	HGreen	**GGreen**	**HGreen**	**GGreen**	**HGreen**	**GGreen**	**HGreen**	**GGreen**
LT101	162	366	928	1069	30	30	1036	1036
LT102	117	265	1009	1162	1138	1160	333	333
LT103	483	1099	1036	1194	972	990	1601	1601
LT104	259	587	234	269	845	862	1161	1161
LT105	123	278	133	153	855	871	1297	1297
LT106	41	93	1005	1157	1302	1327	373	373
LT107	252	572	834	961	410	418	1666	1666
LT108	494	1121	1077	1240	121	123	1416	1416
LT109	390	886	250	288	603	615	1327	1327
LT110	72	164	1019	1173	215	219	1666	1666
LT111	35	79	236	273	1013	1032	61	61
LT112	207	469	520	598	811	826	935	935
LT113	198	448	556	640	1170	1193	610	610
LT114	192	435	740	852	939	957	1260	1260
LT115	494	1121	670	772	182	186	1270	1270
LT116	345	782	391	451	712	725	160	160
LT117	112	254	155	179	1227	1250	1358	1358
LT118	414	941	600	692	452	460	25	25
LT119	394	894	632	727	452	460	1240	1240
LT120	78	177	250	288	328	335	396	396
LT121	354	805	337	389	215	219	836	836
LT122	289	655	172	197	990	1008	647	647
LT123	343	778	868	1000	650	663	847	847
LT124	139	316	515	593	732	746	1388	1388
LT125	68	153	996	1147	1001	1020	75	75
LT126	123	280	661	761	772	787	1006	1006
LT127	409	928	387	446	358	364	1480	1480
LT128	148	336	1102	1269	1355	1381	674	674
LT129	147	334	980	1129	387	394	1188	1188
LT130	259	587	915	1054	834	850	461	461
LT131	36	81	414	476	559	570	1472	1472
LT132	411	932	445	513	454	462	810	810
LT133	454	1031	463	534	317	323	698	698
LT134	441	1002	499	575	761	775	1412	1412
LT135	204	462	165	190	650	663	1226	1226
LT136	252	572	949	1093	1162	1183	1097	1097
LT137	26	59	1002	1155	1346	1371	1368	1368
LT138	360	818	706	814	1068	1089	157	157
LT139	483	1096	23	27	387	394	437	437
LT140	412	934	43	50	442	451	1446	1446
LT141	172	390	704	810	829	844	651	651
LT142	80	182	167	192	241	245	85	85
LT143	447	1015	647	746	855	871	989	989
LT144	148	336	942	1085	1375	1402	935	935
LT145	87	197	567	653	1133	1154	765	765
LT146	311	706	160	184	231	236	1149	1149
LT147	377	856	207	239	1364	1389	1584	1584
LT148	341	773	32	37	1048	1067	495	495
LT149	395	896	1124	1295	1420	1446	765	765
LT150	473	1073	1043	1201	1238	1261	519	519

7.1.6 Análise dos Resultados

Os resultados mostram que a estratégia HGreen produziu consumos de energia menores do que GGreen para todos os cenários avaliados. Considerando o cenário C1, HGreen apresenta uma crescente redução do consumo de energia em relação a GGreen à medida em que se aumenta o número de tarefas. Por exemplo, no lote LT104 com 48 tarefas, HGreen tem consumo cerca de 47% menor que o consumo de GGreen.

Quando se analisa o cenário C2, apesar de os resultados de HGreen serem menores do que os de GGreen, a diferença é menor que a encontrada no cenário C1, sendo inferior a 20% para a maior parte dos lotes de C2. A partir desta observação, pode-se concluir que HGreen tira melhor proveito de cenários menos equilibrados.

Finalmente, os resultados obtidos com os lotes do cenário C3 demonstram o oposto do ocorrido em C1, ou seja, uma decrescente redução do consumo de energia em relação à estratégia GGreen à medida em que se aumenta o número de tarefas. Em outras palavras, os valores dos consumos energéticos de ambas as estratégias aproximam-se quando se aumenta a quantidade de tarefas. Tais valores são totalmente idênticos para todos os lotes com 48 tarefas e divergem por menos de 3% quando consideramos 36 tarefas.

As tabelas apresentadas anteriormente contrapõem os consumos de energia produzidos por HGreen e GGreen. É sabido que seria interessante promover a comparação entre as estratégias aqui propostas e as soluções de escalonamento vigentes em ambientes de grid em produção. Entretanto, essa comparação torna-se difícil uma vez que tais soluções não estão abertas e rodam como uma espécie de caixa preta. Sendo assim, para auxiliar na comparação dos resultados foi desenvolvida uma estratégia de alocação denominada Aleatória.

A estratégia Aleatória consiste em distribuir as tarefas aleatoriamente para os recursos do grid de modo uniforme. Ou seja, os recursos vão “enchendo” de maneira equilibrada até atingirem seu limite de núcleos, deixando assim, a lista de recursos disponíveis. A construção do método de alocação aleatório visa explorar o espaço de soluções, tornando o processo de avaliação das estratégias mais abrangente. Sendo assim, para cada lote de tarefas pertencente aos 3 cenários, foram realizadas 1000 execuções da estratégia de alocação Aleatória. As Tabelas 12 e 13 contêm os resultados de 1000 execuções da estratégia Aleatória para cada lote dos cenários C1 e C2, respectivamente.

Tabela 12: Resultados de 1000 execuções de Aleatória em C1 (kWh)

		12 Taref.	24 Taref.	36 Taref.	48 Taref.			12 Taref.	24 Taref.	36 Taref.	48 Taref.
	Mínimo	649	689	861	1001		Mínimo	430	667	875	1008
LT101	Média	959	1078	1115	1241	*LT126*	Média	847	1050	1161	1227
	Máximo	1101	1153	1226	1327		Máximo	1010	1182	1238	1314
	Mínimo	482	812	816	1003		Mínimo	386	737	941	733
LT102	Média	920	1066	1049	1239	*LT127*	Média	710	1006	1172	1160
	Máximo	1029	1144	1148	1324		Máximo	913	1107	1251	1268
	Mínimo	537	912	900	680		Mínimo	498	704	734	938
LT103	Média	811	1149	1167	1049	*LT128*	Média	978	1034	1148	1163
	Máximo	959	1223	1281	1174		Máximo	1110	1182	1253	1262
	Mínimo	488	713	1011	1082		Mínimo	560	823	857	986
LT104	Média	860	970	1205	1233	*LT129*	Média	986	1095	1132	1218
	Máximo	966	1073	1284	1319		Máximo	1090	1197	1234	1319
	Mínimo	334	780	878	969		Mínimo	445	757	831	1057
LT105	Média	690	1045	1180	1181	*LT130*	Média	885	1032	1072	1249
	Máximo	777	1133	1272	1259		Máximo	1054	1147	1166	1338
	Mínimo	586	675	931	821		Mínimo	529	711	779	1065
LT106	Média	1003	941	1164	1138	*LT131*	Média	1020	1099	1064	1269
	Máximo	1117	1027	1261	1256		Máximo	1107	1188	1151	1340
	Mínimo	495	586	734	965		Mínimo	504	659	778	878
LT107	Média	894	945	1013	1198	*LT132*	Média	905	995	1170	1163
	Máximo	1048	1119	1172	1309		Máximo	1065	1147	1250	1272
	Mínimo	460	855	935	1066		Mínimo	488	701	1009	914
LT108	Média	876	1124	1165	1241	*LT133*	Média	802	1030	1181	1191
	Máximo	969	1195	1247	1336		Máximo	921	1130	1259	1274
	Mínimo	501	793	837	920		Mínimo	510	753	896	911
LT109	Média	883	1076	1105	1133	*LT134*	Média	907	1067	1208	1176
	Máximo	1005	1182	1214	1238		Máximo	1103	1187	1277	1294
	Mínimo	572	629	866	974		Mínimo	491	503	849	876
LT110	Média	958	1003	1137	1235	*LT135*	Média	925	987	1209	1180
	Máximo	1083	1171	1251	1332		Máximo	1045	1104	1281	1301
	Mínimo	678	717	793	1089		Mínimo	633	914	849	1101
LT111	Média	988	1053	1095	1265	*LT136*	Média	900	1123	1114	1255
	Máximo	1092	1137	1185	1351		Máximo	1058	1211	1181	1343
	Mínimo	426	703	835	949		Mínimo	445	592	797	903
LT112	Média	885	944	1065	1206	*LT137*	Média	873	985	1035	1141
	Máximo	1003	1028	1179	1273		Máximo	1049	1139	1114	1233
	Mínimo	479	705	871	869		Mínimo	423	558	869	934
LT113	Média	696	1022	1130	1174	*LT138*	Média	833	991	1171	1202
	Máximo	773	1185	1247	1283		Máximo	915	1106	1265	1309
	Mínimo	417	885	840	1036		Mínimo	472	775	830	989
LT114	Média	822	1152	1118	1231	*LT139*	Média	997	1075	1189	1200
	Máximo	974	1224	1177	1318		Máximo	1117	1160	1264	1302
	Mínimo	513	805	917	1067		Mínimo	523	637	799	900
LT115	Média	959	1052	1185	1244	*LT140*	Média	929	916	1107	1176
	Máximo	1076	1143	1257	1327		Máximo	1022	1012	1186	1277
	Mínimo	393	540	796	794		Mínimo	468	539	848	970
LT116	Média	784	929	1140	1147	*LT141*	Média	939	908	1161	1223
	Máximo	854	1051	1245	1236		Máximo	1107	997	1259	1306
	Mínimo	424	858	844	821		Mínimo	750	705	890	906
LT117	Média	784	1121	1086	1075	*LT142*	Média	1003	1075	1093	1151
	Máximo	926	1209	1173	1192		Máximo	1098	1156	1172	1253
	Mínimo	480	760	770	1023		Mínimo	505	631	909	907
LT118	Média	801	1058	1128	1220	*LT143*	Média	937	1040	1213	1188
	Máximo	992	1176	1245	1318		Máximo	1105	1171	1277	1275
	Mínimo	382	770	906	853		Mínimo	599	536	760	915
LT119	Média	708	992	1132	1112	*LT144*	Média	907	988	1091	1216
	Máximo	906	1085	1240	1198		Máximo	1038	1128	1207	1296
	Mínimo	469	768	841	899		Mínimo	499	843	900	886
LT120	Média	973	1058	1086	1156	*LT145*	Média	840	1098	1178	1171
	Máximo	1101	1147	1163	1258		Máximo	943	1193	1258	1257
	Mínimo	454	646	772	789		Mínimo	436	645	700	941
LT121	Média	904	931	1061	1184	*LT146*	Média	859	1061	1016	1199
	Máximo	1075	1028	1170	1289		Máximo	1028	1183	1143	1303
	Mínimo	545	755	807	982		Mínimo	377	593	885	899
LT122	Média	834	1071	1060	1197	*LT147*	Média	528	863	1142	1172
	Máximo	1006	1174	1148	1284		Máximo	610	962	1253	1264
LT123	Mínimo	468	688	793	957		Mínimo	523	622	961	993
	Média	815	972	1152	1243	*LT148*	Média	927	1049	1170	1217
	Máximo	1013	1088	1228	1318		Máximo	1108	1179	1278	1298
LT124	Mínimo	533	717	841	1035		Mínimo	514	905	949	944
	Média	830	1014	1144	1248	*LT149*	Média	833	1153	1173	1219
	Máximo	926	1134	1233	1337		Máximo	978	1221	1256	1313
LT125	Mínimo	523	636	954	977		Mínimo	454	647	969	789
	Média	991	1057	1153	1171	*LT150*	Média	871	1001	1152	1128
	Máximo	1094	1162	1247	1262		Máximo	1039	1123	1232	1252

Tabela 13: Consumos de energia para 1000 execuções de Aleatória em C2 (kWh)

		12 Taref.	24 Taref.	36 Taref.	48 Taref.			12 Taref.	24 Taref.	36 Taref.	48 Taref.
	Mínimo	452	494	545	680		Mínimo	366	431	605	838
LT101	Média	957	914	1037	1133	*LT126*	Média	732	855	1051	1121
	Máximo	1087	1027	1139	1221		Máximo	816	989	1149	1247
	Mínimo	371	376	699	751		Mínimo	290	469	729	708
LT102	Média	754	625	1041	1108	*LT127*	Média	602	799	1014	1024
	Máximo	897	766	1186	1193		Máximo	696	948	1150	1123
	Mínimo	342	519	498	742		Mínimo	290	469	729	708
LT103	Média	605	996	974	1116	*LT128*	Média	602	799	1014	1024
	Máximo	812	1141	1091	1238		Máximo	696	948	1150	1123
	Mínimo	447	322	780	552		Mínimo	456	355	685	642
LT104	Média	824	654	1099	996	*LT129*	Média	951	633	1041	1076
	Máximo	1065	715	1197	1149		Máximo	1056	745	1128	1194
	Mínimo	348	478	493	717		Mínimo	243	486	459	703
LT105	Média	688	949	1037	1083	*LT130*	Média	488	1002	957	1113
	Máximo	825	1132	1147	1206		Máximo	535	1133	1061	1200
	Mínimo	429	429	722	855		Mínimo	391	476	622	591
LT106	Média	919	911	1097	1139	*LT131*	Média	800	1059	1081	1148
	Máximo	1029	1006	1190	1215		Máximo	936	1149	1186	1237
	Mínimo	352	460	523	510		Mínimo	380	542	560	941
LT107	Média	669	755	809	1037	*LT132*	Média	681	1006	995	1130
	Máximo	836	905	914	1168		Máximo	906	1132	1162	1239
	Mínimo	419	480	481	550		Mínimo	409	539	596	679
LT108	Média	792	981	982	1049	*LT133*	Média	815	995	1027	1064
	Máximo	1001	1114	1102	1140		Máximo	976	1138	1170	1149
	Mínimo	255	524	428	536		Mínimo	440	334	556	519
LT109	Média	526	967	785	906	*LT134*	Média	863	660	962	953
	Máximo	607	1110	962	997		Máximo	1053	772	1078	1167
	Mínimo	319	498	791	676		Mínimo	366	554	501	874
LT110	Média	575	838	1103	1088	*LT135*	Média	727	834	954	1149
	Máximo	761	1043	1182	1207		Máximo	874	941	1135	1224
	Mínimo	411	447	486	780		Mínimo	472	501	440	801
LT111	Média	806	812	954	1096	*LT136*	Média	916	883	855	1118
	Máximo	986	1021	1118	1229		Máximo	1058	1061	1019	1211
	Mínimo	495	486	514	664		Mínimo	457	395	505	732
LT112	Média	897	705	853	1054	*LT137*	Média	995	824	1002	1050
	Máximo	1055	791	958	1168		Máximo	1095	935	1157	1155
	Mínimo	380	456	547	708		Mínimo	314	503	431	550
LT113	Média	750	940	1004	1090	*LT138*	Média	633	903	814	985
	Máximo	910	1077	1095	1198		Máximo	745	1099	888	1161
	Mínimo	249	546	798	773		Mínimo	383	438	532	513
LT114	Média	500	1029	1078	1094	*LT139*	Média	802	918	952	1061
	Máximo	592	1141	1196	1200		Máximo	901	1051	1113	1177
	Mínimo	234	440	559	781		Mínimo	344	490	790	552
LT115	Média	420	809	897	1120	*LT140*	Média	612	966	1110	1048
	Máximo	552	1040	1013	1216		Máximo	818	1050	1192	1196
	Mínimo	325	372	742	525		Mínimo	220	495	411	530
LT116	Média	624	651	1054	1025	*LT141*	Média	433	931	787	941
	Máximo	769	778	1189	1134		Máximo	520	1055	937	1064
	Mínimo	398	433	669	615		Mínimo	387	502	420	612
LT117	Média	854	784	1000	987	*LT142*	Média	827	1004	754	1039
	Máximo	958	1024	1103	1112		Máximo	932	1140	963	1167
	Mínimo	451	621	494	767		Mínimo	405	687	634	742
LT118	Média	953	1010	942	1132	*LT143*	Média	754	992	1066	1126
	Máximo	1085	1112	1096	1221		Máximo	963	1150	1183	1225
	Mínimo	381	466	443	672		Mínimo	398	533	436	575
LT119	Média	710	951	889	1157	*LT144*	Média	778	927	900	1026
	Máximo	913	1082	1026	1233		Máximo	950	1098	1037	1187
	Mínimo	381	406	585	777		Mínimo	455	611	665	744
LT120	Média	758	758	1034	1151	*LT145*	Média	941	958	983	1118
	Máximo	917	947	1155	1228		Máximo	1090	1103	1109	1233
	Mínimo	334	607	577	579		Mínimo	424	746	722	737
LT121	Média	712	1003	1096	997	*LT146*	Média	760	1066	1056	1111
	Máximo	802	1150	1196	1102		Máximo	1007	1143	1146	1186
	Mínimo	290	491	751	668		Mínimo	386	604	762	546
LT122	Média	564	863	1076	1113	*LT147*	Média	695	984	1056	1061
	Máximo	691	985	1143	1193		Máximo	914	1102	1164	1150
	Mínimo	411	535	511	530		Mínimo	453	399	583	509
LT123	Média	809	1036	907	962	*LT148*	Média	954	824	1075	962
	Máximo	984	1139	1048	1086		Máximo	1091	948	1167	1046
	Mínimo	381	510	579	540		Mínimo	393	536	812	485
LT124	Média	687	953	960	997	*LT149*	Média	842	869	1131	939
	Máximo	905	1088	1152	1092		Máximo	940	971	1211	1113
	Mínimo	352	454	631	784		Mínimo	449	415	753	723
LT125	Média	636	886	990	1085	*LT150*	Média	893	864	1081	1003
	Máximo	833	1045	1130	1196		Máximo	1078	991	1179	1153

As tabelas exibidas anteriormente contêm média, mínimo e máximo consumo de energia para 1000 diferentes alocações geradas a partir da distribuição aleatória das tarefas. Os resultados de Aleatória em C3 não foram significativos para a apresentação de média, mínimo e máximo, logo, apenas a média será apresentada mais adiante. A Tabela 14 exibe um comparativo dos valores obtidos por HGreen e as médias com intervalo de confiança obtidas a partir de 1000 amostras de Aleatória.

Tabela 14: HGreen e médias de 1000 execuções de Aleatória em C1 (kWh)

Lote	12 Tarefas		24 Tarefas		36 Tarefas		48 Tarefas	
	HGreen	**Aleatória***	**HGreen**	**Aleatória***	**HGreen**	**Aleatória***	**HGreen**	**Aleatória***
LT101	435	959 (953;966)	452	1078 (1074;1081)	494	1115 (1110;1119)	553	1241 (1238;1244)
LT102	406	920 (915;926)	445	1066 (1064;1069)	464	1049 (1046;1053)	553	1239 (1236;1243)
LT103	380	811 (805;817)	473	1149 (1145;1153)	515	1167 (1162;1172)	493	1049 (1043;1054)
LT104	381	860 (855;865)	421	970 (965;975)	514	1205 (1203;1208)	555	1233 (1230;1235)
LT105	310	690 (684;695)	440	1045 (1041;1048)	513	1180 (1176;1184)	560	1181 (1178;1184)
LT106	442	1003 (996;1009)	402	941 (936;946)	504	1164 (1161;1168)	527	1138 (1133;1143)
LT107	416	894 (887;900)	444	945 (937;953)	476	1013 (1006;1020)	552	1198 (1194;1202)
LT108	381	876 (871;881)	463	1124 (1121;1128)	499	1165 (1162;1168)	589	1241 (1238;1244)
LT109	398	883 (878;888)	462	1076 (1072;1080)	491	1105 (1100;1109)	526	1133 (1129;1136)
LT110	426	958 (952;963)	464	1003 (995;1011)	501	1137 (1132;1141)	560	1235 (1232;1239)
LT111	428	988 (983;993)	444	1053 (1050;1057)	478	1095 (1091;1099)	569	1265 (1262;1267)
LT112	398	885 (880;891)	398	944 (941;947)	474	1065 (1061;1070)	528	1206 (1204;1209)
LT113	304	696 (692;699)	465	1022 (1015;1029)	501	1130 (1125;1135)	536	1174 (1170;1178)
LT114	389	822 (815;829)	475	1152 (1149;1156)	473	1118 (1115;1120)	607	1231 (1228;1233)
LT115	425	959 (954;965)	445	1052 (1048;1056)	506	1185 (1182;1189)	568	1244 (1241;1247)
LT116	338	784 (779;788)	417	929 (924;935)	501	1140 (1136;1145)	518	1147 (1142;1151)
LT117	370	784 (776;791)	473	1121 (1117;1126)	472	1086 (1082;1090)	504	1075 (1070;1080)
LT118	396	801 (793;810)	461	1058 (1053;1063)	504	1128 (1123;1134)	554	1220 (1216;1223)
LT119	366	708 (697;719)	426	992 (988;996)	500	1132 (1127;1136)	504	1112 (1108;1116)
LT120	441	973 (965;980)	447	1058 (1055;1062)	469	1086 (1083;1089)	523	1156 (1153;1160)
LT121	431	904 (896;912)	403	931 (927;935)	474	1061 (1056;1066)	541	1184 (1180;1189)
LT122	399	834 (827;841)	458	1071 (1066;1076)	462	1060 (1056;1065)	536	1197 (1193;1200)
LT123	404	815 (806;824)	429	972 (967;977)	493	1152 (1149;1156)	552	1243 (1240;1246)
LT124	365	830 (825;835)	446	1014 (1009;1020)	494	1144 (1141;1147)	569	1248 (1245;1251)
LT125	434	991 (985;996)	456	1057 (1051;1063)	502	1153 (1149;1157)	598	1171 (1168;1175)
LT126	404	847 (838;855)	465	1050 (1044;1056)	497	1161 (1157;1165)	560	1227 (1224;1230)
LT127	367	710 (700;720)	434	1006 (1002;1011)	500	1172 (1169;1176)	528	1160 (1154;1165)
LT128	441	978 (971;985)	465	1034 (1027;1040)	503	1148 (1144;1152)	532	1163 (1160;1166)
LT129	427	986 (981;990)	468	1095 (1091;1099)	498	1132 (1127;1136)	555	1218 (1215;1222)
LT130	423	885 (877;893)	450	1032 (1028;1037)	468	1072 (1068;1076)	564	1249 (1246;1252)
LT131	434	1020 (1016;1024)	467	1099 (1094;1104)	460	1064 (1060;1067)	558	1269 (1267;1272)
LT132	424	905 (897;912)	453	995 (988;1002)	502	1170 (1166;1174)	536	1163 (1159;1167)
LT133	364	802 (797;807)	442	1030 (1025;1034)	506	1181 (1177;1184)	543	1191 (1188;1194)
LT134	440	907 (898;916)	468	1067 (1061;1073)	512	1208 (1205;1211)	545	1176 (1172;1181)
LT135	416	925 (917;932)	435	987 (982;991)	513	1209 (1205;1213)	556	1180 (1175;1185)
LT136	417	900 (894;906)	473	1123 (1119;1126)	475	1114 (1111;1117)	566	1255 (1252;1257)
LT137	422	873 (864;883)	450	985 (978;993)	448	1035 (1032;1038)	514	1141 (1137;1144)
LT138	363	833 (828;837)	436	991 (985;997)	511	1171 (1166;1175)	550	1202 (1198;1207)
LT139	443	997 (990;1005)	454	1075 (1072;1079)	509	1189 (1186;1193)	544	1200 (1197;1203)
LT140	400	929 (925;934)	396	916 (911;921)	474	1107 (1103;1110)	534	1176 (1173;1180)
LT141	441	939 (931;946)	390	908 (904;912)	505	1161 (1157;1166)	547	1223 (1219;1227)
LT142	429	1003 (999;1007)	452	1075 (1071;1079)	471	1093 (1090;1096)	530	1151 (1148;1155)
LT143	441	937 (929;944)	463	1040 (1034;1047)	513	1213 (1210;1216)	537	1188 (1185;1191)
LT144	408	907 (902;912)	443	988 (982;994)	487	1091 (1086;1096)	542	1216 (1213;1219)
LT145	373	840 (835;846)	465	1098 (1094;1102)	502	1178 (1174;1181)	534	1171 (1167;1175)
LT146	413	859 (850;869)	465	1061 (1054;1068)	459	1016 (1011;1021)	546	1199 (1195;1203)
LT147	241	528 (525;532)	381	863 (858;868)	503	1142 (1138;1147)	536	1172 (1168;1176)
LT148	441	927 (919;935)	464	1049 (1042;1056)	515	1170 (1166;1173)	540	1217 (1214;1220)
LT149	388	833 (826;839)	474	1153 (1150;1156)	502	1173 (1170;1176)	550	1219 (1215;1223)
LT150	414	871 (863;879)	439	1001 (995;1006)	495	1152 (1149;1155)	526	1128 (1122;1134)

* Média e o Intervalo de Confiança de 95%

Similarmente, a Tabela 15 traz o mesmo comparativo para os lotes do cenário C2. Conforme pode ser verificado nas Tabelas 14 e 15, os consumos de energia retornados por HGreen estão bem abaixo da média obtida pela distribuição aleatória, considerando os cenários C1 e C2.

Tabela 15: HGreen e médias de 1000 execuções de Aleatória em C2 (kWh)

Lote	12 Tarefas		24 Tarefas		36 Tarefas		48 Tarefas	
	HGreen	**Aleatória***	**HGreen**	**Aleatória***	**HGreen**	**Aleatória***	**HGreen**	**Aleatória***
LT101	437	957 (948;967)	407	914 (908;920)	450	1037 (1032;1041)	494	1133 (1128;1137)
LT102	365	754 (745;762)	308	625 (618;631)	473	1041 (1035;1047)	487	1108 (1104;1112)
LT103	336	605 (593;617)	452	996 (989;1002)	436	974 (967;980)	501	1116 (1111;1120)
LT104	434	824 (811;837)	284	654 (650;658)	477	1099 (1095;1104)	472	996 (989;1003)
LT105	333	688 (681;695)	452	949 (941;958)	462	1037 (1030;1045)	493	1083 (1078;1089)
LT106	413	919 (912;925)	397	911 (906;916)	469	1097 (1092;1102)	494	1139 (1135;1142)
LT107	340	669 (660;678)	362	755 (748;762)	368	809 (803;815)	476	1037 (1030;1044)
LT108	409	792 (780;803)	441	981 (974;988)	442	982 (975;988)	462	1049 (1044;1054)
LT109	245	526 (521;531)	441	967 (960;973)	390	785 (776;794)	408	906 (902;910)
LT110	312	575 (564;585)	419	838 (828;847)	466	1103 (1100;1107)	488	1088 (1083;1093)
LT111	400	806 (796;815)	411	812 (802;823)	451	954 (946;962)	500	1096 (1090;1101)
LT112	421	897 (889;904)	313	705 (702;709)	386	853 (847;859)	476	1054 (1048;1060)
LT113	367	750 (742;757)	432	940 (931;949)	433	1004 (999;1008)	486	1090 (1085;1094)
LT114	237	500 (495;504)	451	1029 (1023;1036)	476	1078 (1073;1082)	484	1094 (1090;1098)
LT115	226	420 (412;427)	425	809 (795;822)	405	897 (891;904)	491	1120 (1116;1125)
LT116	313	624 (616;632)	312	651 (645;657)	473	1054 (1048;1060)	461	1025 (1019;1030)
LT117	386	854 (848;861)	417	784 (770;797)	442	1000 (994;1006)	452	987 (982;992)
LT118	438	953 (945;960)	439	1010 (1004;1015)	438	942 (935;950)	494	1132 (1129;1136)
LT119	373	710 (699;721)	433	951 (944;959)	416	889 (881;898)	506	1157 (1153;1162)
LT120	374	758 (748;767)	384	758 (748;767)	460	1034 (1027;1040)	494	1151 (1147;1154)
LT121	324	712 (706;719)	456	1003 (997;1010)	473	1096 (1091;1101)	450	997 (992;1003)
LT122	280	564 (558;570)	391	863 (856;869)	449	1076 (1073;1079)	483	1113 (1109;1116)
LT123	394	809 (801;817)	452	1036 (1030;1042)	420	907 (900;913)	444	962 (956;968)
LT124	372	687 (674;700)	429	953 (948;959)	463	960 (951;970)	442	997 (993;1001)
LT125	345	636 (624;647)	422	886 (877;895)	452	990 (984;997)	487	1085 (1080;1090)
LT126	326	732 (728;737)	392	855 (849;861)	457	1051 (1046;1055)	507	1121 (1117;1126)
LT127	281	602 (596;609)	379	799 (793;806)	459	1014 (1009;1020)	456	1024 (1020;1029)
LT128	422	602 (596;609)	298	799 (793;806)	447	1014 (1009;1020)	483	1024 (1020;1029)
LT129	213	951 (945;958)	452	633 (628;638)	425	1041 (1036;1046)	483	1076 (1070;1083)
LT130	377	488 (485;491)	451	1002 (995;1008)	471	957 (951;963)	502	1113 (1108;1117)
LT131	372	800 (793;807)	446	1059 (1054;1065)	466	1081 (1076;1085)	504	1148 (1144;1152)
LT132	394	681 (668;694)	454	1006 (1000;1012)	471	995 (988;1003)	464	1130 (1127;1134)
LT133	428	815 (807;823)	308	995 (988;1001)	433	1027 (1019;1034)	478	1064 (1060;1068)
LT134	351	863 (853;872)	375	660 (655;666)	456	962 (956;969)	494	953 (943;963)
LT135	422	727 (720;734)	423	834 (828;839)	415	954 (945;963)	488	1149 (1145;1153)
LT136	440	916 (909;922)	371	883 (875;892)	463	855 (846;864)	472	1118 (1113;1123)
LT137	299	995 (988;1001)	441	824 (818;829)	357	1002 (994;1009)	478	1050 (1044;1055)
LT138	360	633 (628;639)	415	903 (893;913)	448	814 (809;819)	479	985 (976;993)
LT139	337	802 (796;807)	412	918 (912;923)	470	952 (944;960)	490	1061 (1054;1067)
LT140	211	612 (601;624)	422	966 (962;971)	383	1110 (1107;1114)	434	1048 (1041;1055)
LT141	376	433 (429;437)	450	931 (923;939)	393	787 (779;795)	474	941 (936;946)
LT142	391	827 (819;835)	456	1004 (998;1010)	471	754 (741;766)	492	1039 (1033;1046)
LT143	384	754 (743;766)	436	992 (985;999)	417	1066 (1061;1072)	487	1126 (1121;1131)
LT144	438	778 (770;786)	438	927 (919;934)	441	900 (893;906)	499	1026 (1018;1033)
LT145	415	941 (933;948)	446	958 (951;964)	457	983 (978;989)	479	1118 (1114;1123)
LT146	379	760 (745;774)	434	1066 (1062;1070)	463	1056 (1051;1061)	466	1111 (1108;1115)
LT147	438	695 (682;708)	380	984 (979;989)	463	1056 (1050;1061)	426	1061 (1056;1066)
LT148	378	954 (947;961)	383	824 (815;832)	479	1075 (1070;1081)	456	962 (957;967)
LT149	436	842 (836;848)	395	869 (864;874)	469	1131 (1126;1135)	477	939 (932;947)
LT150	401	893 (884;902)		864 (857;871)	439	1081 (1076;1086)	490	1003 (997;1009)

* Média e o Intervalo de Confiança de 95%

A Tabela 16 exibe os resultados de GGreen com as médias obtidas a partir de 1000 execuções Aleatória em C1.

Tabela 16: GGreen e médias de 1000 execuções de Aleatória em C1 (kWh)

Lote	12 Tarefas		24 Tarefas		36 Tarefas		12 Tarefas	
	GGreen	**Aleatória***	**GGreen**	**Aleatória***	**GGreen**	**Aleatória***	**GGreen**	**Aleatória***
LT101	458	959 (953;966)	565	1078 (1074;1081)	725	1115 (1110;1119)	947	1241 (1238;1244)
LT102	463	920 (915;926)	503	1066 (1064;1069)	706	1049 (1046;1053)	1008	1239 (1236;1243)
LT103	391	811 (805;817)	611	1149 (1145;1153)	862	1167 (1162;1172)	622	1049 (1043;1054)
LT104	422	860 (855;865)	486	970 (965;975)	940	1205 (1203;1208)	1048	1233 (1230;1235)
LT105	350	690 (684;695)	521	1045 (1041;1048)	841	1180 (1176;1184)	913	1181 (1178;1184)
LT106	512	1003 (996;1009)	493	941 (936;946)	768	1164 (1161;1168)	836	1138 (1133;1143)
LT107	429	894 (887;900)	473	945 (937;953)	611	1013 (1006;1020)	930	1198 (1194;1202)
LT108	443	876 (871;881)	609	1124 (1121;1128)	810	1165 (1162;1168)	1048	1241 (1238;1244)
LT109	415	883 (878;888)	527	1076 (1072;1080)	830	1105 (1100;1109)	903	1133 (1129;1136)
LT110	466	958 (952;963)	519	1003 (995;1011)	662	1137 (1132;1141)	1000	1235 (1232;1239)
LT111	488	988 (983;993)	625	1053 (1050;1057)	724	1095 (1091;1099)	1069	1265 (1262;1267)
LT112	433	885 (880;891)	578	944 (941;947)	741	1065 (1061;1070)	848	1206 (1204;1209)
LT113	344	696 (692;699)	540	1022 (1015;1029)	763	1130 (1125;1135)	858	1174 (1170;1178)
LT114	400	822 (815;829)	755	1152 (1149;1156)	750	1118 (1115;1120)	1003	1231 (1228;1233)
LT115	464	959 (954;965)	586	1052 (1048;1056)	770	1185 (1182;1189)	1064	1244 (1241;1247)
LT116	391	784 (779;788)	453	929 (924;935)	763	1140 (1136;1145)	763	1147 (1142;1151)
LT117	383	784 (776;791)	630	1121 (1117;1126)	658	1086 (1082;1090)	786	1075 (1070;1080)
LT118	406	801 (793;810)	545	1058 (1053;1063)	726	1128 (1123;1134)	908	1220 (1216;1223)
LT119	374	708 (697;719)	642	992 (988;996)	793	1132 (1127;1136)	826	1112 (1108;1116)
LT120	492	973 (965;980)	668	1058 (1055;1062)	664	1086 (1083;1089)	851	1156 (1153;1160)
LT121	442	904 (896;912)	473	931 (927;935)	700	1061 (1056;1066)	754	1184 (1180;1189)
LT122	411	834 (827;841)	664	1071 (1066;1076)	616	1060 (1056;1065)	954	1197 (1193;1200)
LT123	415	815 (806;824)	516	972 (967;977)	686	1152 (1149;1156)	888	1243 (1240;1246)
LT124	422	830 (825;835)	489	1014 (1009;1020)	794	1144 (1141;1147)	1052	1248 (1245;1251)
LT125	485	991 (985;996)	536	1057 (1051;1063)	795	1153 (1149;1157)	995	1171 (1168;1175)
LT126	415	847 (838;855)	518	1050 (1044;1056)	810	1161 (1157;1165)	1015	1227 (1224;1230)
LT127	376	710 (700;720)	582	1006 (1002;1011)	853	1172 (1169;1176)	757	1160 (1154;1165)
LT128	483	978 (971;985)	501	1034 (1027;1040)	765	1148 (1144;1152)	920	1163 (1160;1166)
LT129	476	986 (981;990)	526	1095 (1091;1099)	770	1132 (1127;1136)	1031	1218 (1215;1222)
LT130	434	885 (877;893)	482	1032 (1028;1037)	642	1072 (1068;1076)	1026	1249 (1246;1252)
LT131	505	1020 (1016;1024)	541	1099 (1094;1104)	634	1064 (1060;1067)	1011	1269 (1267;1272)
LT132	437	905 (897;912)	477	995 (988;1002)	725	1170 (1166;1174)	862	1163 (1159;1167)
LT133	385	802 (797;807)	540	1030 (1025;1034)	960	1181 (1177;1184)	912	1191 (1188;1194)
LT134	452	907 (898;916)	564	1067 (1061;1073)	838	1208 (1205;1211)	879	1176 (1172;1181)
LT135	472	925 (917;932)	474	987 (982;991)	780	1209 (1205;1213)	907	1180 (1175;1185)
LT136	431	900 (894;906)	536	1123 (1119;1126)	810	1114 (1111;1117)	1068	1255 (1252;1257)
LT137	433	873 (864;883)	481	985 (978;993)	697	1035 (1032;1038)	779	1141 (1137;1144)
LT138	412	833 (828;837)	496	991 (985;997)	815	1171 (1166;1175)	904	1202 (1198;1207)
LT139	514	997 (990;1005)	545	1075 (1072;1079)	774	1189 (1186;1193)	988	1200 (1197;1203)
LT140	455	929 (925;934)	514	916 (911;921)	722	1107 (1103;1110)	893	1176 (1173;1180)
LT141	453	939 (931;946)	598	908 (904;912)	748	1161 (1157;1166)	960	1223 (1219;1227)
LT142	480	1003 (999;1007)	523	1075 (1071;1079)	726	1093 (1090;1096)	860	1151 (1148;1155)
LT143	453	937 (929;944)	498	1040 (1034;1047)	910	1213 (1210;1216)	927	1188 (1185;1191)
LT144	428	907 (902;912)	472	988 (982;994)	656	1091 (1086;1096)	866	1216 (1213;1219)
LT145	423	840 (835;846)	752	1098 (1094;1102)	787	1178 (1174;1181)	865	1171 (1167;1175)
LT146	424	859 (850;869)	548	1061 (1054;1068)	652	1016 (1011;1021)	918	1199 (1195;1203)
LT147	261	528 (525;532)	446	863 (858;868)	705	1142 (1138;1147)	910	1172 (1168;1176)
LT148	454	927 (919;935)	541	1049 (1042;1056)	881	1170 (1166;1173)	959	1217 (1214;1220)
LT149	400	833 (826;839)	652	1153 (1150;1156)	754	1173 (1170;1176)	885	1219 (1215;1223)
LT150	425	871 (863;879)	599	1001 (995;1006)	875	1152 (1149;1155)	703	1128 (1122;1134)

* Média e o Intervalo de Confiança de 95%

A Tabela 17 contém os consumos de energia de GGreen para os lotes do cenário C2. Como pode ser verificado, a estratégia GGreen também apresenta resultados

melhores que as médias encontradas dentre as 1000 execuções de Aleatória. Novamente para C3 não houve resultados significativos.

Tabela 17: GGreen e médias de 1000 execuções de Aleatória em C2 (kWh)

Lote	12 Tarefas		24 Tarefas		36 Tarefas		48 Tarefas	
	GGreen	Aleatória*	GGreen	Aleatória*	GGreen	Aleatória*	GGreen	Aleatória*
LT101	506	957 (948;967)	445	914 (908;920)	498	1037 (1032;1041)	567	1133 (1128;1137)
LT102	384	754 (745;762)	318	625 (618;631)	496	1041 (1035;1047)	559	1108 (1104;1112)
LT103	339	605 (593;617)	468	996 (989;1002)	493	974 (967;980)	739	1116 (1111;1120)
LT104	442	824 (811;837)	330	654 (650;658)	542	1099 (1095;1104)	522	996 (989;1003)
LT105	342	688 (681;695)	466	949 (941;958)	517	1037 (1030;1045)	545	1083 (1078;1089)
LT106	468	919 (912;925)	442	911 (906;916)	551	1097 (1092;1102)	870	1139 (1135;1142)
LT107	347	669 (660;678)	372	755 (748;762)	403	809 (803;815)	546	1037 (1030;1044)
LT108	416	792 (780;803)	494	981 (974;988)	477	982 (975;988)	541	1049 (1044;1054)
LT109	265	526 (521;531)	464	967 (960;973)	407	785 (776;794)	453	906 (902;910)
LT110	316	575 (564;585)	430	838 (828;847)	537	1103 (1100;1107)	538	1088 (1083;1093)
LT111	409	806 (796;815)	421	812 (802;823)	469	954 (946;962)	676	1096 (1090;1101)
LT112	432	897 (889;904)	345	705 (702;709)	431	853 (847;859)	553	1054 (1048;1060)
LT113	375	750 (742;757)	482	940 (931;949)	498	1004 (999;1008)	630	1090 (1085;1094)
LT114	246	500 (495;504)	524	1029 (1023;1036)	532	1078 (1073;1082)	597	1094 (1090;1098)
LT115	230	420 (412;427)	434	809 (795;822)	454	897 (891;904)	757	1120 (1116;1125)
LT116	320	624 (616;632)	322	651 (645;657)	496	1054 (1048;1060)	528	1025 (1019;1030)
LT117	445	854 (848;861)	425	784 (770;797)	506	1000 (994;1006)	499	987 (982;992)
LT118	487	953 (945;960)	509	1010 (1004;1015)	476	942 (935;950)	679	1132 (1129;1136)
LT119	380	710 (699;721)	484	951 (944;959)	447	889 (881;898)	589	1157 (1153;1162)
LT120	382	758 (748;767)	394	758 (748;767)	495	1034 (1027;1040)	741	1151 (1147;1154)
LT121	374	712 (706;719)	481	1003 (997;1010)	537	1096 (1091;1101)	584	997 (992;1003)
LT122	286	564 (558;570)	425	863 (856;869)	611	1076 (1073;1079)	823	1113 (1109;1116)
LT123	404	809 (801;817)	506	1036 (1030;1042)	440	907 (900;913)	480	962 (956;968)
LT124	376	687 (674;700)	444	953 (948;959)	484	960 (951;970)	497	997 (993;1001)
LT125	348	636 (624;647)	435	886 (877;895)	542	990 (984;997)	540	1085 (1080;1090)
LT126	359	732 (728;737)	404	855 (849;861)	514	1051 (1046;1055)	804	1121 (1117;1126)
LT127	315	602 (596;609)	390	799 (793;806)	570	1014 (1009;1020)	523	1024 (1020;1029)
LT128	478	602 (596;609)	308	799 (793;806)	515	1014 (1009;1020)	564	1024 (1020;1029)
LT129	246	951 (945;958)	496	633 (628;638)	487	1041 (1036;1046)	608	1076 (1070;1083)
LT130	398	488 (485;491)	523	1002 (995;1008)	523	957 (951;963)	618	1113 (1108;1117)
LT131	376	800 (793;807)	489	1059 (1054;1065)	486	1081 (1076;1085)	869	1148 (1144;1152)
LT132	403	681 (668;694)	468	1006 (1000;1012)	513	995 (988;1003)	533	1130 (1127;1134)
LT133	436	815 (807;823)	318	995 (988;1001)	487	1027 (1019;1034)	500	1064 (1060;1068)
LT134	360	863 (853;872)	415	660 (655;666)	477	962 (956;969)	699	953 (943;963)
LT135	460	727 (720;734)	436	834 (828;839)	431	954 (945;963)	754	1149 (1145;1153)
LT136	508	916 (909;922)	404	883 (875;892)	485	855 (846;864)	694	1118 (1113;1123)
LT137	308	995 (988;1001)	454	824 (818;829)	416	1002 (994;1009)	500	1050 (1044;1055)
LT138	398	633 (628;639)	435	903 (893;913)	467	814 (809;819)	549	985 (976;993)
LT139	341	802 (796;807)	477	918 (912;923)	578	952 (944;960)	513	1061 (1054;1067)
LT140	216	612 (601;624)	479	966 (962;971)	399	1110 (1107;1114)	517	1048 (1041;1055)
LT141	434	433 (429;437)	476	931 (923;939)	407	787 (779;795)	525	941 (936;946)
LT142	398	827 (819;835)	470	1004 (998;1010)	517	754 (741;766)	574	1039 (1033;1046)
LT143	393	754 (743;766)	448	992 (985;999)	435	1066 (1061;1072)	542	1126 (1121;1131)
LT144	469	778 (770;786)	452	927 (919;934)	467	900 (893;906)	636	1026 (1018;1033)
LT145	420	941 (933;948)	520	958 (951;964)	536	983 (978;989)	632	1118 (1114;1123)
LT146	382	760 (745;774)	477	1066 (1062;1070)	527	1056 (1051;1061)	545	1111 (1108;1115)
LT147	469	695 (682;708)	430	984 (979;989)	542	1056 (1050;1061)	497	1061 (1056;1066)
LT148	427	954 (947;961)	436	824 (815;832)	697	1075 (1070;1081)	476	962 (957;967)
LT149	446	842 (836;848)	429	869 (864;874)	547	1131 (1126;1135)	615	939 (932;947)
LT150	411	893 (884;902)	466	864 (857;871)	506	1081 (1076;1086)	563	1003 (997;1009)

* Média e o Intervalo de Confiança de 95%

A Tabela 18 exibe os resultados de HGreen, GGreen e as médias de Aleatória em relação ao cenário C3. Novamente, HGreen esteve abaixo da média de todos os lotes. Destaca-se também o fato de que à medida que o número de tarefas aumenta (aproximando-se da capacidade total dos recursos) as estratégias convergem para o mesmo resultado.

Tabela 18: HGreen, GGreen e médias de 1000 execuções de Aleatória em C3

Lote	12 Tarefas			24 Tarefas			36 Tarefas			48 Tarefas		
	HGreen	GGreen	Aleatória	HGreen	GGreen	Aleatória	HGreen	GGreen	Aleatória	HGreen	GGreen	Aleatória
LT101	*162*	*366*	*411*	*928*	*1069*	*1165*	*30*	*30*	*32*	*1036*	*1036*	*1036*
LT102	*117*	*265*	*298*	*1009*	*1162*	*1266*	*1138*	*1160*	*1237*	*333*	*333*	*333*
LT103	*483*	*1099*	*1234*	*1036*	*1194*	*1300*	*972*	*990*	*1056*	*1601*	*1601*	*1601*
LT104	*259*	*587*	*660*	*234*	*269*	*293*	*845*	*862*	*919*	*1161*	*1161*	*1161*
LT105	*123*	*278*	*313*	*133*	*153*	*167*	*855*	*871*	*928*	*1297*	*1297*	*1297*
LT106	*41*	*93*	*104*	*1005*	*1157*	*1261*	*1302*	*1327*	*1415*	*373*	*373*	*373*
LT107	*252*	*572*	*642*	*834*	*961*	*1047*	*410*	*418*	*446*	*1666*	*1666*	*1666*
LT108	*494*	*1121*	*1258*	*1077*	*1240*	*1351*	*121*	*123*	*131*	*1416*	*1416*	*1416*
LT109	*390*	*886*	*994*	*250*	*288*	*314*	*603*	*615*	*655*	*1327*	*1327*	*1327*
LT110	*72*	*164*	*184*	*1019*	*1173*	*1278*	*215*	*219*	*233*	*1666*	*1666*	*1666*
LT111	*35*	*79*	*89*	*236*	*273*	*297*	*1013*	*1032*	*1101*	*61*	*61*	*61*
LT112	*207*	*469*	*527*	*520*	*598*	*652*	*811*	*826*	*881*	*935*	*935*	*935*
LT113	*198*	*448*	*504*	*556*	*640*	*697*	*1170*	*1193*	*1272*	*610*	*610*	*610*
LT114	*192*	*435*	*488*	*740*	*852*	*928*	*939*	*957*	*1021*	*1260*	*1260*	*1260*
LT115	*494*	*1121*	*1258*	*670*	*772*	*841*	*182*	*186*	*198*	*1270*	*1270*	*1270*
LT116	*345*	*782*	*879*	*391*	*451*	*491*	*712*	*725*	*773*	*160*	*160*	*160*
LT117	*112*	*254*	*285*	*155*	*179*	*195*	*1227*	*1250*	*1333*	*1358*	*1358*	*1358*
LT118	*414*	*941*	*1057*	*600*	*692*	*753*	*452*	*460*	*490*	*25*	*25*	*25*
LT119	*394*	*894*	*1005*	*632*	*727*	*793*	*452*	*460*	*490*	*1240*	*1240*	*1240*
LT120	*78*	*177*	*200*	*250*	*288*	*314*	*328*	*335*	*357*	*396*	*396*	*396*
LT121	*354*	*805*	*904*	*337*	*389*	*424*	*215*	*219*	*233*	*836*	*836*	*836*
LT122	*289*	*655*	*735*	*172*	*197*	*215*	*990*	*1008*	*1075*	*647*	*647*	*647*
LT123	*343*	*778*	*874*	*868*	*1000*	*1089*	*650*	*663*	*707*	*847*	*847*	*847*
LT124	*139*	*316*	*355*	*515*	*593*	*647*	*732*	*746*	*796*	*1388*	*1388*	*1388*
LT125	*68*	*153*	*172*	*996*	*1147*	*1249*	*1001*	*1020*	*1088*	*75*	*75*	*75*
LT126	*123*	*280*	*315*	*661*	*761*	*830*	*772*	*787*	*840*	*1006*	*1006*	*1006*
LT127	*409*	*928*	*1042*	*387*	*446*	*486*	*358*	*364*	*388*	*1480*	*1480*	*1480*
LT128	*148*	*336*	*378*	*1102*	*1269*	*1383*	*1355*	*1381*	*1473*	*674*	*674*	*674*
LT129	*147*	*334*	*375*	*980*	*1129*	*1230*	*387*	*394*	*420*	*1188*	*1188*	*1188*
LT130	*259*	*587*	*660*	*915*	*1054*	*1148*	*834*	*850*	*906*	*461*	*461*	*461*
LT131	*36*	*81*	*91*	*414*	*476*	*520*	*559*	*570*	*608*	*1472*	*1472*	*1472*
LT132	*411*	*932*	*1047*	*445*	*513*	*559*	*454*	*462*	*493*	*810*	*810*	*810*
LT133	*454*	*1031*	*1158*	*463*	*534*	*581*	*317*	*323*	*344*	*698*	*698*	*698*
LT134	*441*	*1002*	*1125*	*499*	*575*	*627*	*761*	*775*	*827*	*1412*	*1412*	*1412*
LT135	*204*	*462*	*519*	*165*	*190*	*207*	*650*	*663*	*707*	*1226*	*1226*	*1226*
LT136	*252*	*572*	*642*	*949*	*1093*	*1190*	*1162*	*1183*	*1262*	*1097*	*1097*	*1097*
LT137	*26*	*59*	*66*	*1002*	*1155*	*1258*	*1346*	*1371*	*1463*	*1368*	*1368*	*1368*
LT138	*360*	*818*	*919*	*706*	*814*	*886*	*1068*	*1089*	*1161*	*157*	*157*	*157*
LT139	*483*	*1096*	*1231*	*23*	*27*	*29*	*387*	*394*	*420*	*437*	*437*	*437*
LT140	*412*	*934*	*1050*	*43*	*50*	*54*	*442*	*451*	*481*	*1446*	*1446*	*1446*
LT141	*172*	*390*	*438*	*704*	*810*	*884*	*829*	*844*	*900*	*651*	*651*	*651*
LT142	*80*	*182*	*204*	*167*	*192*	*209*	*241*	*245*	*261*	*85*	*85*	*85*
LT143	*447*	*1015*	*1140*	*647*	*746*	*813*	*855*	*871*	*929*	*989*	*989*	*989*
LT144	*148*	*336*	*378*	*942*	*1085*	*1182*	*1375*	*1402*	*1495*	*935*	*935*	*935*
LT145	*87*	*197*	*222*	*567*	*653*	*711*	*1133*	*1154*	*1231*	*765*	*765*	*765*
LT146	*311*	*706*	*793*	*160*	*184*	*200*	*231*	*236*	*252*	*1149*	*1149*	*1149*
LT147	*377*	*856*	*961*	*207*	*239*	*260*	*1364*	*1389*	*1482*	*1584*	*1584*	*1584*
LT148	*341*	*773*	*868*	*32*	*37*	*40*	*1048*	*1067*	*1139*	*495*	*495*	*495*
LT149	*395*	*896*	*1007*	*1124*	*1295*	*1410*	*1420*	*1446*	*1543*	*765*	*765*	*765*

A seguir, as Figuras 9, 10 e 11 exibem gráficos que expressam os consumos de energia para HGreen, GGreen e os mínimos obtidos da estratégia Aleatória considerando a série de lotes com 36 tarefas para os respectivos cenários C1, C2 e C3.

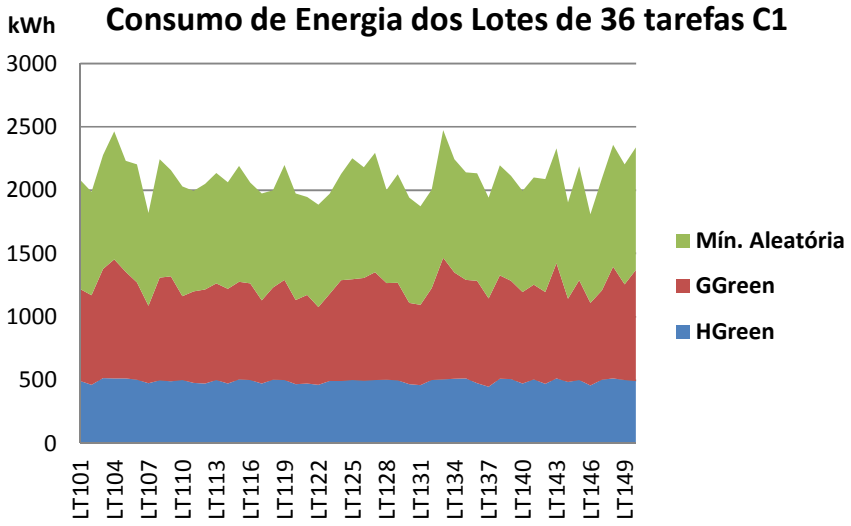

Figura 9: Gráfico dos resultados de HGreen, GGreen e Mín. Aleatória em C1

Os gráficos destacam e confirmam o menor consumo obtido por HGreen em todos os cenários avaliados. Além disso, evidencia que nehuma das 1000 alocações obtidas pela distribuição aleatória conseguiu gerar um consumo menor que ambas as estratégias aqui propostas.

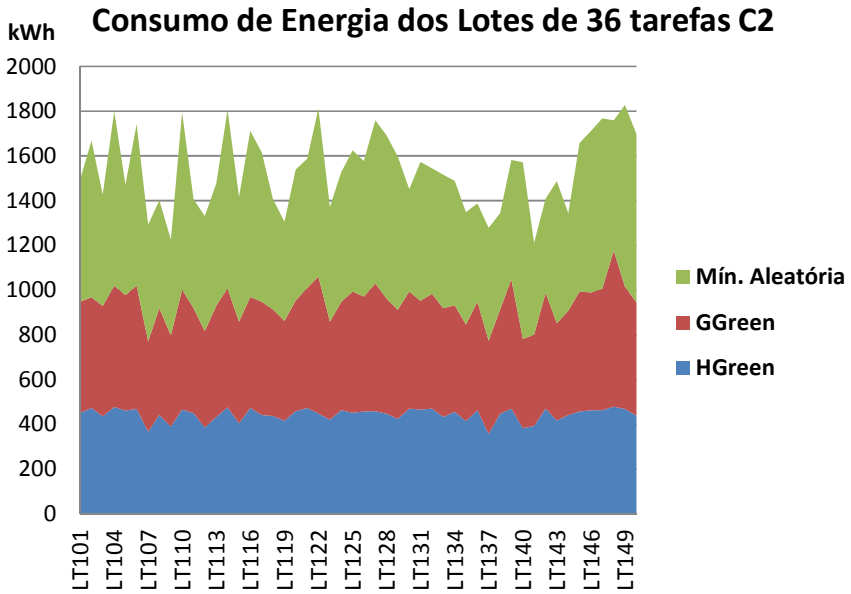

Figura 10: Gráfico dos resultados de HGreen, GGreen e Mín. Aleatória em C2

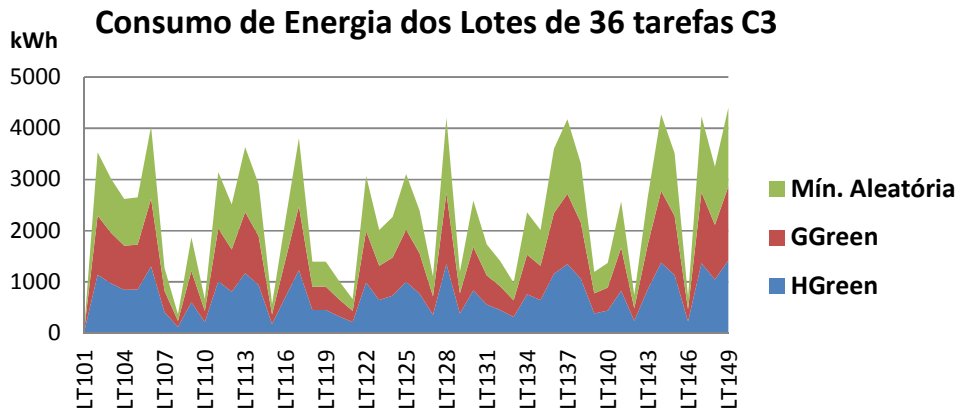

Figura 11: Gráfico dos resultados de HGreen, GGreen e Mín. Aleatória em C3

Esta seção apresentou os consumos de energia das estratégias HGreen e GGreen, uma vez que tais estratégias possuem como meta principal a redução dos mesmos. No Apêndice 1, podem ser encontrados os tempos de execução retornados pelas estratégias HGreen e GGreen para todos os cenários avaliados. A estratégia BOTEEN será analisada separadamente na seção seguinte.

7.2 Resultados de BOTEEN

Esta seção apresenta os resultados obtidos na avaliação da estratégia BOTEEN, a qual foi descrita na seção 6.3. A fim de implementar o algoritmo BOTEEN foi desenvolvido um programa em linguagem C (código-fonte no Apêndice 4).

Para a análise da estratégia foram construídos 3 cenários distintos com 200 tarefas cada. Dessa forma, os cenários denominados BC1, BC2 e BC3 possuem tarefas distribuídas similarmente aos cenários da seção anterior. Ou seja, de elevada variância (BC1); baixa variância (BC2); e variância nula (BC3). Os pesos das tarefas são dados em operações de ponto flutuante para adequar-se às métricas do *benchmark* utilizado.

A Tabela 19 apresenta informações sobre as 24 máquinas utilizadas como base para a execução de BOTEEN frente aos cenários descritos anteriormente. As máquinas da foram selecionadas da lista Green500 (GREEN500, [S.d.]) e apesar de possuírem um número elevado de núcleos, para esta avaliação, foi considerado que cada máquina teria 16 núcleos disponíveis para execução.

Tabela 19: Configuração dos recursos usados na análise de BOTEEN

Ranque Green500	Mflops/W por Núcleo	Descrição	Num. Núcleos	Gflops por Núcleo
1	0,263864583333	BlueGene/Q 1.60 GHz	7680	22,460156250000
5	0,145627369872	NNSA/SC Blue Gene/Q P1	11604	5,631420199931
6	0,139224242424	DEGIMA Cluster, Intel i5	9900	6,203030303030
10	0,008984415514	HP ProLiant Xeon 6C X5670	73278	16,266819509266
44	0,017976308267	Cray XE6 Opteron 2.10 GHz	36384	6,519349164468
45	0,024111842105	Amazon EC2 Cluster 2.60GHz	17024	14,103031015038
75	0,028086805556	iDataPlex DX360M3, Xeon 2.66	14400	9,465277777778
81	0,054571759259	Power 775 3.836 GHz	6912	23,090277777778
134	0,050626795977	HS22, Xeon QC GT 2.66 GHz	5568	9,214089439655
149	0,001129049527	Cray XT5-HE Opteron 2.6 GHz	224162	7,847003506393
172	0,019339110644	HS22 Xeon E5649 6C 2.53 GHz	11424	5,635066526611
187	0,018029059829	HS22 Xeon X5650 6C 2.66 GHz	11700	5,626102564103
208	0,006416005291	iDataPlex, Xeon E55xx 2.53 GHz	30240	5,575396825397
233	0,016009134781	x3650M3, Xeon X56xx 2.53 GHz	11604	5,635039641503
244	0,013910315274	x3550M3 Xeon X5650 2.66 GHz	13068	5,635062748699
275	0,006744221411	Sun R422, Xeon X5570, 2.93 GHz	26304	10,447080291971
333	0,003348005244	Cray XE6 8-core 2.4 GHz	42712	7,873665480427
359	0,014541294643	x3650M2 Xeon E55xx 2.53 Ghz	8960	5,714657366071
378	0,009424398625	x3650M2 Xeon E55xx 2.26 Ghz	11640	4,806250000000
386	0,002351555137	Sun x6275, Xeon X55xx 2.93 Ghz	42440	10,214420358153
413	0,002132537688	Cray XT3/XT4	38208	5,344430485762
488	0,003115170380	eServer pSeries p5 575 1.9 GHz	12208	6,205766710354
496	0,001641698473	Cray XT5 QC 2.4 GHz	20960	7,900763358779
500	0,002367021277	PowerEdge 1850, 3.6 GHz	9024	5,873226950355

As métricas Gflops e Mflops/W permitem que sejam previamente calculados os tempos de execução e os consumos de energia, respectivamente. Dessa forma, para cada uma das 200 máquinas é calculado tempo e energia considerando os 24 recursos disponíveis. Esses dados (não exibidos por questões de espaço) são carregados em uma matriz e servem de entrada para o programa BOTEEN.

A Tabela 20 apresenta os resultados (tempo/energia em minutos e kWh, respectivamente) encontrados para as 96 alocações Pareto-ótimas geradas para o conjunto-solução do cenário BC1.

Tabela 20: Resultados de BOTEEN para o cenário BC1

Cenário BC1							
Sol	**Tempo/Energia**	**Sol**	**Tempo/Energia**	**Sol**	**Tempo/Energia**	**Sol**	**Tempo/Energia**
1	2042/211506	25	1361/226665	49	1154/252677	73	933/373004
2	1983/211555	26	1356/226674	50	1145/252696	74	923/387532
3	1924/211562	27	1290/227476	51	1128/254841	75	917/387623
4	1908/211587	28	1284/227737	52	1102/259921	76	916/387643
5	1894/211682	29	1273/230487	53	1095/263957	77	914/402371
6	1881/211745	30	1268/230499	54	1065/263971	78	898/417743
7	1854/211872	31	1266/231367	55	1049/263997	79	888/428695
8	1827/211998	32	1263/232088	56	1048/264493	80	887/428711
9	1805/213329	33	1250/232160	57	1036/267760	81	868/432036
10	1800/213357	34	1248/232912	58	1031/267782	82	863/432666
11	1776/214086	35	1243/235122	59	1021/268233	83	858/443177
12	1773/214105	36	1236/235131	60	1013/273970	84	850/443206
13	1746/215690	37	1233/236365	61	1006/274286	85	845/444018
14	1720/218495	38	1230/237050	62	1004/274299	86	839/468001
15	1639/220206	39	1215/238809	63	995/290482	87	833/473164
16	1628/221285	40	1213/239951	64	994/290843	88	828/478680
17	1612/221319	41	1209/239971	65	986/297742	89	827/499631
18	1585/221730	42	1197/241222	66	977/312945	90	814/503405
19	1558/222269	43	1194/243476	67	968/313179	91	806/503721
20	1532/223262	44	1182/245369	68	967/329350	92	804/512074
21	1505/223941	45	1180/246227	69	959/354903	93	799/517834
22	1478/224584	46	1176/247341	70	951/355128	94	796/517858
23	1421/225880	47	1162/250092	71	941/366699	95	792/517903
24	1397/225914	48	1155/251187	72	940/366834	96	789/529335

Considerando os cenários BC1 e BC2, a escolha da alocação Pareto-ótimo a ser utilizada necessitaria de um método para tomada de decisão. Sob o olhar da computação verde, um possível procedimento seria normalizar os resultados dos consumos de energia e aplicar um parâmetro similar ao *fator verde*, definido na seção 6.1.

A Tabela 21 reúne os resultados obtidos a partir das 70 alocações Pareto-ótimas geradas por BOTEEN para o cenário BC2.

Tabela 21: Resultados de BOTEEN para o cenário BC2

Cenário BC2							
Sol	**Tempo/Energia**	**Sol**	**Tempo/Energia**	**Sol**	**Tempo/Energia**	**Sol**	**Tempo/Energia**
1	2123/309502	19	1397/351363	37	1233/375456	55	1092/481645
2	2042/309629	20	1370/356767	38	1230/375812	56	1085/492930
3	2015/309851	21	1358/358519	39	1222/376443	57	1056/497944
4	1988/310074	22	1357/358959	40	1215/384001	58	1049/522075
5	1935/310581	23	1356/359140	41	1212/385679	59	1021/523630
6	1908/310898	24	1343/361472	42	1206/386085	60	1018/535842
7	1881/311722	25	1339/365285	43	1197/388914	61	1013/551366
8	1854/312040	26	1321/365420	44	1180/390561	62	1009/555951
9	1773/312640	27	1310/366328	45	1173/390926	63	989/555970
10	1720/313176	28	1303/367223	46	1171/398182	64	986/556550
11	1693/313673	29	1302/367788	47	1162/398595	65	977/585773
12	1666/314657	30	1285/368060	48	1158/402922	66	959/587825
13	1612/317101	31	1284/368935	49	1140/404279	67	957/588496
14	1558/326962	32	1268/370152	50	1133/407218	68	951/589476
15	1505/329113	33	1266/372029	51	1127/421229	69	941/600040
16	1451/343474	34	1250/372796	52	1122/450773	70	933/600485
17	1424/346684	35	1248/374155	53	1121/450776		
18	1414/351089	36	1245/375012	54	1109/452643		

E, finalmente, para o cenário BC3, cujas tarefas possuem o mesmo peso, foram encontradas apenas 4 alocações Pareto-ótimas, retornando os seguintes dados: 89/63208; 87/183096; 85/231576 e 81/244512.

Os resultados encontrados evidenciam que para um lote com tarefas de pesos idênticos o número de soluções obtidas é bastante baixo, permitindo que a tomada de decisão possa ser feita por alguém responsável pela execução.

Capítulo 8 – Conclusão

Nos últimos anos, a computação em grid tem se consolidado como uma solução capaz de integrar, em escala global, recursos geograficamente distribuídos e heterogêneos. Tal fato tem contribuído de maneira significativa para o aumento da infraestrutura de TI, proporcionando a utilização de uma vasta malha computacional, a qual demanda um elevado consumo de energia para seu funcionamento.

De fato, dados recentes têm apontado que a indústria de tecnologia da informação e comunicação é responsável por aproximadamente 2% da emissão global de CO_2, valor equivalente a toda a indústria de aviação (GARTNER NEWSROOM, 2007). Dessa forma, o consumo de energia pela área de TI tem gerado preocupações não apenas no aspecto financeiro, mas principalmente em relação ao impacto ambiental acarretado.

Tal realidade tem motivado a comunidade científica a buscar o desenvolvimento de tecnologias que permitam o crescimento de maneira sustentável. Neste contexto, o presente trabalho propõe algumas estratégias para a alocação de recursos em ambientes de grid. O estudo desse tema rendeu contribuições que serão detalhadas na seção a seguir.

8.1 Contribuições

Possivelmente, o principal valor deste trabalho tenha sido contribuir para que os usuários do grid possam utilizar os recursos disponíveis de uma forma mais sustentável. Todavia, algumas contribuições mais pontuais precisam ser citadas, o que é feito a seguir:

- Análise comparativa das soluções de escalonamento em grid existentes na literatura;
- Definição de um modelo de dados para a representação do consumo de energia em ambientes de grid;
- Aplicar políticas verdes a partir de métodos não intrusivos, diferentemente dos trabalhos baseados na técnica DVS;
- Criação das estratégias HGreen e GGreen através da definição de heurísticas para a computação verde;

- Análise do problema de alocação aplicado sobre o contexto da otimização multiobjetivo com o intuito de reduzir energia e tempo (Algoritmo BOTEEN);
- Implementação das estratégias propostas e análise de testes realizados através de simuladores de grid ou via desenvolvimento de software;
- Estudo e extensão da ferramenta de simulação GridSim para seu uso aplicado à computação verde.

O resultado dos estudos citados anteriormente, além de outros trabalhos de pesquisa ao longo do curso, produziram as seguintes publicações:

- *Many Task Computing for Orthologous Genes Identification in Protozoan Genomes using Hydra*. **Fábio Coutinho**, Ogasawara, E., Oliveira, D., Braganholo, V., Lima, A. A., Alberto M. R. Dávila, Marta Mattoso. International Journal Concurrency and Computation: Practice & Experience, 2011.
- *A Workflow Scheduling Algorithm for Optimizing Energy-Efficient Grid Resources Usage. Fábio Coutinho, Renato Santana, Luís Alfredo V. de Carvalho. IEEE International Conference on Cloud and Green Computing, 2011.*
- *Data Parallelism in Bioinformatics Workflows Using Hydra In: Workshop Emerging Computational Methods for the Life Sciences. Fábio Coutinho, Ogasawara, E., Oliveira, D., Braganholo, V., Lima, A. A., Alberto M. R. Dávila, Marta Mattoso. ACM International Symposium on High Performance Distributed Computing, 2010.*

Para conquistar tais resultados, muitas dificuldades foram encontradas ao longo do caminho. A seção seguinte sintetiza os principais obstáculos enfrentados no decorrer da tese.

8.2 Dificuldades

A computação em grid e a computação verde tratam de assuntos de cobertura ampla. Tal propriedade, nos estudos e pesquisas iniciais, dificultou em muito a convergência para um tema específico.

Além disso, a computação verde é uma área bastante recente, o que seria salutar do ponto de vista do ineditismo, mas que impõe o desafio de caminhar sobre tópicos ainda pouco consolidados. Um exemplo disso foi a dificuldade em estimar o consumo total da energia gasta por um computador para executar determinada tarefa, uma vez que existe muita informação sobre o consumo de processadores, mas o mesmo não ocorre para outros dispositivos envolvidos na execução (por exemplo: disco, dispositivos de rede, etc.).

Outra resultante da recentidade do tema foi encontrada nos simuladores de grid que ainda não se encontram preparados para simular o consumo de energia dos recuros. Tal fato obrigou a adaptação do simulador utilizado para promover os testes das estratégias de alocação.

8.3 Trabalhos Futuros

Esta seção descreve possíveis extensões deste trabalho, bem como traça novos caminhos a serem seguidos.

Considerando o atual estágio do trabalho, algumas perspectivas futuras são listadas a seguir:

- Avaliar outros cenários de execução com o intuito de investigar novas realidades;
- Promover estudo sobre novas estratégias verdes de alocação, bem como o aprimoramento das estratégias já definidas;
- Continuar buscando a convergência dos aspectos da computação verde com os objetivos tradicionais que cercam o problema da alocação de tarefas em grids;
- Analisar as estratégias propostas a partir de um ambiente de grid real, incluindo a adição de medidores do consumo de energia aos equipamentos;
- Considerar a aplicação das estratégias de alocação em outras plataformas de computação distribuída, tal como *nuvem*;
- Estender a implementação visando atender a outras características do modelo de energia proposto;

- Estudar alternativas para estender o modelo atual a fim de tratar outras fontes de consumo de energia (refrigeração, dispositivos de comunicação, etc.).

Na seção seguinte são encontradas as últimas considerações que definitivamente encerram este trabalho.

8.4 Considerações Finais

Os resultados da simulação mostraram que a estratégia HGreen superou GGreen em todos os cenários avaliados. Além disso, ambas as estratégias foram mais eficientes que as amostras geradas a partir da distribuição aleatória de tarefas.

A execução do cenário C3 confirmou que as estratégias HGreen e GGreen exercem pouca influência sobre a alocação de lotes com tarefas homogêneas. Por outro lado, em cenário equivalente, a estratégia BOTEEN mostrou-se plenamente viável, retornando um número pequeno de alocações Pareto-ótimas.

Portanto, os triunfos deste trabalho demonstram que é possível utilizar recursos do grid consumindo menos energia. E mais ainda, que isso pode ser feito sem desconsiderar por inteira a questão do desempenho, ou seja, pomovendo soluções que considerem ambos os objetivos de minimizar tempo e energia.

Por fim, ficou claro no decorrer dos estudos para esta tese que a sustentabilidade ambiental apresenta-se como um tema de natureza interdisciplinar e ainda com muitas questões em aberto. Contudo, uma relevante conclusão que se pode tirar, por mais óbvia que pareça, está apoiada em um conhecido pensamento: *é preciso que cada um faça sua a parte*.

Referências Bibliográficas

ABRAMSON, D., SOSIC, R., GIDDY, J., *et al.* "Nimrod: A Tool for Performing Parametised Simulations using Distributed Workstations". In: **Proceedings of the 4th IEEE Symposium on High Performance Distributed Computing**, pp. 112-121, Virginia, Ago. 1995.

ALBAYRAKTAROGLU, K., JALEEL, A., WU, X., *et al.* "BioBench: A Benchmark Suite of Bioinformatics Applications". In: **Proceedings of the IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS 2005)**, pp. 2-9, Austin, Texas, USA, Mar. 2005.

ALLEN, G., ANGULO, D., GOODALE, T., *et al.* "GridLab: Enabling Applications on the Grid". In: **Proceedings of the 3rd International Workshop on Grid Computing, held in conjunction with Supercomputing 2002**, v. 2536, pp. 39-45, Baltimore, MD, USA, Nov. 2002.

AMD. *AMD Cool'n'Quiet Technology*. [S.l: s.n.]: [S.d.]. Disponível em: <http://www.amd.com/us/products/technologies/cool-n-quiet/Pages/cool-n-quiet.aspx>. Acesso em: 1 nov. 2011.

AMIN, K., LASZEWSKI, G. VON, HATEGAN, M., *et al.* "GridAnt: a client-controllable grid workflow system". In: **Proceedings of the 37th Annual Hawaii International Conference on System Sciences (HICSS 2004) - Track 7**, v. 7, pp. 70210c, Hawaii, USA, Jan. 2004.

APACHE. *The Apache ANT Project*. [S.l: s.n.]: [S.d.]. Disponível em: <http://ant.apache.org/>. Acesso em: 30 dez. 2011.

BADER, D. A., YUE LI, TAO LI, *et al.* "BioPerf: A Benchmark Suite to Evaluate High-Performance Computer Architecture on Bioinformatics Applications". In: **Proceedings of the IEEE International Workload Characterization Symposium (IISWC 2005)**, pp. 163- 173, Austin, Texas, USA, Out. 2005.

BARHAM, P., DRAGOVIC, B., FRASER, K., *et al.* "Xen and the art of virtualization". In: **Proceedings of the nineteenth ACM Symposium on Operating Systems Principles (SOSP 2003)**, pp. 164–177, New York, NY, Out. 2003.

BAVOIL, L., CALLAHAN, S. P., CROSSNO, P. J., *et al.* "VisTrails: enabling interactive multiple-view visualizations". In: **Proceedings of the IEEE Visualization 2005 (VIS 05)**, pp. 135- 142, Out. 2005.

BELHAJJAME, K., EMBURY, S. M., PATON, N. W., *et al.* "Automatic annotation of Web services based on workflow definitions", **ACM Transactions on the Web**, v. 2, n. 2, pp. 1-34, 2008a.

BELHAJJAME, K., WOLSTENCROFT, K., CORCHO, O., *et al.* "Metadata Management in the Taverna Workflow System". In: **Eighth IEEE International Symposium on Cluster Computing and the Grid (CCGRID 2008)**, pp. 651-656, Los Alamitos, CA, USA, 2008b.

BELOGLAZOV, A., BUYYA, R. "Energy Efficient Resource Management in Virtualized Cloud Data Centers". In: **Proceedings of the 10th IEEE/ACM International Conference on Cluster Cloud and Grid Computing (CCGrid 2010)**, pp. 826-831, Melbourne, Victoria, Mai. 2010.

BERMAN, O., EINAV, D., HANDLER, G. "The Constrained Bottleneck Problem in Networks", **Operations Research**, v. 38, n. 1, pp. 178–181, Fev. 1990.

BHAGAT, J., TANOH, F., NZUOBONTANE, E., *et al.* "BioCatalogue: a universal catalogue of web services for the life sciences", **Nucleic Acids Research**, v. 38, n. Web Server, pp. 689-694, 2010.

BINNEY, A. *Making The Web A Better Place: Guidelines For "Green" Web Design - Smashing Magazine*. [S.l: s.n.]: 2010. Disponível em: <http://www.smashingmagazine.com/2010/09/20/making-the-web-a-better-place-guidelines-for-green-web-design/>. Acesso em: 1 nov. 2011.

BLYTHE, J., JAIN, S., DEELMAN, E., *et al.* "Task scheduling strategies for workflow-based applications in grids". In: **Proceedings of the IEEE International Symposium on Cluster Computing and the Grid (CCGrid 2005)**, v. 2, pp. 759- 767, Cardiff, UK, Mai. 2005.

BORNSTEIN, C. T., MACULAN, N., PASCOAL, M., *et al.* "Multiobjective combinatorial optimization problems with a cost and several bottleneck objective functions: An algorithm with reoptimization", **Computers & Operations Research**, v. 39, n. 9, pp. 1969-1976, Set. 2012.

BRANDIC, I., BENKNER, S., ENGELBRECHT, G., *et al.* "QoS Support for Time-Critical Grid Workflow Applications". In: **Proceedings of the First International Conference on e-Science and Grid Computing (E-SCIENCE 2005)**, pp. 108-115, Melbourne, Australia, Dez. 2005.

BROWN, R., MASANET, E., NORDMAN, B., *et al. Report to congress on server and data center energy efficiency Public law 109-431*. In: Technical Report LBNL-363E, U.S. Environmental Protection Agency ENERGY STAR Program, 2007.

BUYYA, R., ABRAMSON, D., GIDDY, J. "A Case for Economy Grid Architecture for Service-Oriented Grid Computing". In: **Proceedings of the 10th IEEE International Heterogeneous Computing Workshop (HCW 2001)**, v. 2, pp. 776-790, California, USA, Abr. 2001.

BUYYA, R., BELOGLAZOV, A., ABAWAJY, J. "Energy-Efficient Management of Data Center Resources for Cloud Computing: A Vision, Architectural Elements, and Open Challenges". In: **Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA 2010)**, v. 1, pp. 6-20, Las Vegas, Nevada, Jul. 2010.

BUYYA, R., MURSHED, M. "GridSim: A Toolkit for the Modeling and Simulation of Distributed Resource Management and Scheduling for Grid Computing", **Concurrency and Computation: Practice and Experience**, v. 14, n. 13, pp. 1175-1220, 2002.

BUYYA, R., VENUGOPAL, S. "The Gridbus Toolkit for Service Oriented Grid and

Utility Computing: An Overview and Status Report". In: **Proceedings of the 1st IEEE International Workshop on Grid Economics and Business Models (GECON 2004)**, pp. 19- 66, Seoul, South-Korea, Abr. 2004.

BUYYA, R., VENUGOPAL, S. "A Gentle Introduction to Grid Computing and Technologies", **Computer Society of India (CSI) Communications**, v. 29, n. 1, pp. 9-19, Jul. 2005.

CAPPELLO, F., CARON, E., DAYDE, M., *et al.* "Grid'5000: a large scale and highly reconfigurable grid experimental testbed". In: **Proceedings of the 6th IEEE/ACM International Workshop on Grid Computing**, pp. 1-8, Seattle, USA, Nov. 2005.

CARDOSO, F. **Efeito Estufa: Por que a Terra Morre de Calor**. 1a. ed. Terceiro Nome, 2006.

CASANOVA, H. "Simgrid: A Toolkit for the Simulation of Application Scheduling". In: **Proceedings of the First IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid 2001)**, pp. 430-437, Brisbane, Australia, Mai. 2001.

CERN. *Guide to CERN Computing Center*. [S.l: s.n.]: [S.d.]. Disponível em: <http://malandes.web.cern.ch/malandes/cc.html>. Acesso em: 3 jun. 2012.

CPU2006. *SPEC CPU2006*. [S.l: s.n.]: [S.d.]. Disponível em: <http://www.spec.org/cpu2006/>. Acesso em: 16 jan. 2012.

DECEMBER/2011. *EGI Availability/Reliability Reports for December 2011*. [S.l: s.n.]: [S.d.]. Disponível em: <https://documents.egi.eu/public/ShowDocument?docid=959>. Acesso em: 29 jan. 2012.

DEELMAN, E., BLYTHE, J., GIL, A., *et al.* "Pegasus: Mapping scientific workflows onto the grid". In: **Proceedings of the 2nd European Across Grids Conference**, pp. 11-20, Cyprus, 2004a.

DEELMAN, E., BLYTHE, J., GIL, Y., *et al.* "Workflow Management in GriPhyN". In: Nabrzyski, J., Schopf, J. M., Weglarz, J. (eds), **Grid resource management**, Chapter 1, Norwell, MA, USA, Kluwer Academic Publishers, 2004b.

DIRAC. *DIRAC LHCb-Production*. [S.l: s.n.]: [S.d.]. Disponível em: <https://lhcbweb.pic.es/DIRAC/LHCb-Production/user/jobs/SiteSummary/display>. Acesso em: 12 mar. 2012.

DONGARRA, J. J., LUSZCZEK, P., PETITET, A. "The LINPACK benchmark: past, present, and future", **Concurrency and Computation: Practice and Experience**, v. 15, n. 9, pp. 803-820, 2003.

DOWNEY, A. B. "Using Queue Time Predictions for Processor Allocation". In: **Proceedings of the 3rd Workshop on Job Scheduling Strategies for Parallel Processing in conjunction with IPPS 1997**, v. 1291, pp. 35-57, Geneva, Switzerland, Abr. 1997.

EGI. *European Grid Infrastructure (EGI)*. [S.l: s.n.]: [S.d.]. Disponível em:

<http://www.egi.eu/>. Acesso em: 2 jan. 2012.

FAHRINGER, T., PRODAN, R., DUAN, R., *et al.* "ASKALON: A Development and Grid Computing Environment for Scientific Workflows". In: Taylor, I. J., Deelman, E., Gannon, D. B., *et al.* (eds), **Workflows for e-Science**, Part III, London, Springer Verlag, 2007.

FOSTER, I., KESSELMAN, C. **The Grid: Blueprint for a New Computing Infrastructure**. 1st. ed. Massachusetts, Morgan Kaufmann, 1998.

FOSTER, I., KESSELMAN, C., TUECKE, S. "The Anatomy of the Grid: Enabling Scalable Virtual Organizations", **International Journal of High Performance Computing Applications**, v. 15, n. 3, pp. 200-222, 2001.

FOSTER, I., VOCKLER, J., WILDE, M., *et al.* "Chimera: a virtual data system for representing, querying, and automating data derivation". In: **Proceedings of the 14th International Conference on Scientific and Statistical Database Management (SSDBM 2002))**, pp. 37- 46, Edinburgh, Scotland, Jul. 2002.

FREY, J., TANNENBAUM, T., LIVNY, M., *et al.* "Condor-G: a computation management agent for multi-institutional grids". In: **Proceedings of the 10th IEEE International Symposium on High Performance Distributed Computing**, pp. 55-63, San Francisco, CA , USA, Ago. 2001.

GARG, S., BUYYA, R. "Exploiting Heterogeneity in Grid Computing for Energy-Efficient Resource Allocation". In: **Proceedings of the 17th International Conference on Advanced Computing and Communications (ADCOM 2009)**, Bengaluru, India, Dez. 2009.

GARTNER. *Gartner Inc.* [S.l: s.n.]: [S.d.]. Disponível em: <http://www.gartner.com>. Acesso em: 14 fev. 2012.

GARTNER NEWSROOM. *Gartner Estimates ICT Industry Accounts for 2 Percent of Global CO2 Emissions*. [S.l: s.n.]: 2007. Disponível em: <http://www.gartner.com/it/page.jsp?id=503867>. Acesso em: 14 fev. 2012.

GLITE. *gLite - Lightweight Middleware for Grid Computing*. [S.l: s.n.]: [S.d.]. Disponível em: <http://glite.cern.ch/>. Acesso em: 21 nov. 2011.

GLOBUS. *The Globus Alliance*. [S.l: s.n.]: [S.d.]. Disponível em: <http://www.globus.org/>. Acesso em: 21 nov. 2011.

GREEN500. *The Green500 List*. [S.l: s.n.]: [S.d.]. Disponível em: <http://www.green500.org/>. Acesso em: 1 nov. 2011.

GRIDBUS. *Gridbus Project*. [S.l: s.n.]: [S.d.]. Disponível em: <http://www.cloudbus.org/middleware/>. Acesso em: 30 dez. 2011.

GRMS. *Grid(Lab) Resource Management*. [S.l: s.n.]: [S.d.]. Disponível em: <http://www.gridlab.org/WorkPackages/wp-9/>. Acesso em: 29 dez. 2011.

HEP-SPEC06. *HEP-SPEC06 Benchmark*. [S.l: s.n.]: [S.d.]. Disponível em:

<http://w3.hepix.org/benchmarks/doku.php?id=homepage>. Acesso em: 16 jan. 2012.

HOTTA, Y., SATO, M., KIMURA, H., *et al.* "Profile-based Optimization of Power Performance by using Dynamic Voltage Scaling on a PC cluster". In: **Proceedings of the 20th IEEE International Parallel and Distributed Processing Symposium (IPDPS 2006)**, pp. 8, Rhodes Island, Greece, Abr. 2006.

HOWELL, F., MCNAB, R. "SimJava: A discrete event simulation library for java", **Simulation Series**, v. 30, pp. 51-56, 1998.

HSU, C., FENG, W. "A Power-Aware Run-Time System for High-Performance Computing". In: **Proceedings of the ACM/IEEE Conference on Supercomputing (SC 2005)**, pp. 1–9, Seattle, USA, Nov. 2005.

INTEL. *Energy Efficient Servers with Intel® Xeon® Processors*. [S.l: s.n.]: [S.d.]. Disponível em: <http://www.intel.com/content/www/us/en/data-center/energy-efficient-servers-with-xeon-processors.html>. Acesso em: 1 nov. 2011.

IOSUP, A., EPEMA, D. "GRENCHMARK: A Framework for Analyzing, Testing, and Comparing Grids". In: **Proceedings of the 6th IEEE International Symposium on Cluster Computing and the Grid (CCGRID 2006)**, v. 1, pp. 313-320, Singapore, Mai. 2006.

IOZONE. *IOzone Filesystem Benchmark*. [S.l: s.n.]: [S.d.]. Disponível em: <http://www.iozone.org/>. Acesso em: 18 jan. 2012.

JEJURIKAR, R., GUPTA, R. "Energy-Aware Task Scheduling With Task Synchronization for Embedded Real-Time Systems", **IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems**, v. 25, n. 6, pp. 1024-1037, Jun. 2006.

JIN, H., HUANG, J., XIE, X., *et al.* "JFreeSim: a grid simulation tool based on MTMSMR model". In: **Proceedings of the 6th International Conference on Advanced Parallel Processing Technologies (APTT 2005)**, pp. 332–341, Hong Kong, China, 2005.

KAPPIAH, N., FREEH, V. W., LOWENTHAL, D. K. "Just In Time Dynamic Voltage Scaling: Exploiting Inter-Node Slack to Save Energy in MPI Programs". In: **Proceedings of the ACM/IEEE Conference on Supercomputing (SC 2005)**, pp. 1-9, Seattle, USA, Nov. 2005.

KOMMINENI, J., ABRAMSON, D. "GriddLeS Enhancements and Building Virtual Applications for the GRID with Legacy Components". In: **Proceedings of the Advances in Grid Computing - EGC 2005, European Grid Conference**, pp. 961-971, Amsterdam, The Netherlands, Fev. 2005.

KWOK, Y.-K., AHMAD, I. "Static scheduling algorithms for allocating directed task graphs to multiprocessors", **ACM Computing Surveys**, v. 31, n. 4, pp. 406-471, Dez. 1999.

KYONG, H. K., BUYYA, R., JONG, K. "Power Aware Scheduling of Bag-of-Tasks Applications with Deadline Constraints on DVS-enabled Clusters". In: **Proceedings of**

the 7th IEEE International Symposium on Cluster Computing and the Grid (CCGRID 2007), pp. 541-548, Rio de Janeiro, Brazil, Mai. 2007.

LAMEHAMEDI, H., ZUJUN SHENTU, SZYMANSKI, B., *et al.* "Simulation of dynamic data replication strategies in Data Grids". In: **Proceedings of the Parallel and Distributed Processing Symposium in Heterogeneous Computing Workshop (HCW 2003)**, pp. 1-10, Nice, France, Abr. 2003.

LAMMIE, M., BRENNER, P., THAIN, D. "Scheduling Grid workloads on multicore clusters to minimize energy and maximize performance". In: **Proceedings of the 10th IEEE/ACM International Conference on Grid Computing**, pp. 145-152, Banff, Canada, Out. 2009.

LAWSON, C. L., HANSON, R. J., KINCAID, D. R., *et al.* "Basic Linear Algebra Subprograms for Fortran Usage", **ACM Transactions on Mathematical Software**, v. 5, n. 3, pp. 308–323, Set. 1979.

LEPING, W., YING, L. "Efficient Power Management of Heterogeneous Soft Real-Time Clusters". In: **Proceedings of the Real-Time Systems Symposium (RTSS 2008)**, pp. 323-332, Barcelona, Spain, Dez. 2008.

LHC. *LHC Project*. [S.l: s.n.]: [S.d.]. Disponível em: <http://lhc.web.cern.ch/lhc/>. Acesso em: 2 jan. 2012.

LIGO. *Advanced Laser Interferometer Gravitational-Wave Observatory*. [S.l: s.n.]: 2000. Disponível em: <https://www.advancedligo.mit.edu/>. Acesso em: 29 out. 2011.

LINGRAND, D., MONTAGNAT, J., GLATARD, T. "Modeling the Latency on Production Grids with Respect to the Execution Context". In: **Proceedings of the 8th IEEE International Symposium on Cluster Computing and the Grid (CCGRID 2008)**, pp. 753-758, Lyon, France, Mai. 2008.

LUDÄSCHER, B., ALTINTAS, I., BERKLEY, C., *et al.* "Scientific workflow management and the Kepler system", **Concurrency and Computation: Practice and Experience**, v. 18, n. 10, pp. 1039-1065, Ago. 2006.

MANDAL, N., DEELMAN, E., MEHTA, G., *et al.* "Integrating Existing Scientific Workflow Systems : The Kepler/Pegasus Example". In: **Proceedings of the 2nd Workshop on Workflows in Support of Large-Scale Science in conjunction with HPDC 2007**, pp. 21–28, Monterey Bay California, USA, Jun. 2007.

MCGOUGH, S., YOUNG, L., AFZAL, A., *et al.* "Workflow Enactment in ICENI". In: **Proceedings of the UK e-Science All Hands Meeting**, pp. 894-900, Nottingham, UK, Set. 2004.

MEARIAN, L. *MySpace replaces all server hard disks with flash drives - Computerworld*. [S.l: s.n.]: 2009. Disponível em: <http://www.computerworld.com/s/article/9139280/MySpace_replaces_all_server_hard_disks_with_flash_drives>. Acesso em: 1 nov. 2011.

MEINHARD, H. "Power Efficiency of Servers". In: **Proceedings of the HEPiX Fall 2008**, Taipei, Taiwan, Out. 2008.

MEISNER, D., GOLD, B. T., WENISCH, T. F. "PowerNap: eliminating server idle power". In: **Proceedings of the 14th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS 2009)**, pp. 205–216, Washington, USA, Mar. 2009.

MINGAY, S. *10 Key Elements of a "Green IT" Strategy*. In: Technical Report G00154085, Gartner, Inc, Dez. 2007.

MURUGESAN, S. "Harnessing Green IT: Principles and Practices", **IT Professional**, v. 10, n. 1, pp. 24-33, 2008.

MYGRID. *myGrid Project*. [S.l: s.n.]: [S.d.]. Disponível em: <http://www.mygrid.org.uk/>. Acesso em: 26 dez. 2011.

NAQVI, S., RIGUIDEL, M. "Grid Security Services Simulator (G3S) - a simulation tool for the design and analysis of grid security solutions". In: **Proceedings of the First International Conference on e-Science and Grid Computing (e-Science 2005)**, pp. 421-428, Melbourne, Australia, Jul. 2005.

OINN, T., GREENWOOD, M., ADDIS, M., *et al.* "Taverna: Lessons in creating a workflow environment for the life sciences", **Concurrency and Computation Practice and Experience**, v. 18, n. 10, pp. 1067-1100, 2006.

ORGERIE, A.-C., LEFEVRE, L., GELAS, J.-P. "Save Watts in Your Grid: Green Strategies for Energy-Aware Framework in Large Scale Distributed Systems". In: **Proceedings of the 14th IEEE International Conference on Parallel and Distributed Systems (ICPADS 2008)**, pp. 171-178, Melbourne, Australia, Dez. 2008a.

ORGERIE, A.-C., LEFEVRE, L., GELAS, J.-P. "Chasing Gaps between Bursts: Towards Energy Efficient Large Scale Experimental Grids". In: **Proceedings of the Ninth International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT 2008)**, pp. 381-389, Dunedin, New Zealand, Dez. 2008b.

PEREZ, J., GERMAIN-RENAUD, C., KEGL, B., *et al.* "Utility-Based Reinforcement Learning for Reactive Grids". In: **Proceedings of the 5th IEEE International Conference on Autonomic Computing (ICAC 2008)**, pp. 205-206, Chicago, Illinois, USA, Jun. 2008.

PREECE, A., MISSIER, P., EMBURY, S., *et al.* "An ontology-based approach to handling information quality in e-Science", **Concurrency and Computation: Practice and Experience**, v. 20, n. 3, pp. 253-264, 10 Mar. 2008.

ROSA, L., SANTOS, M., MATVIENKO, B., *et al.* "Hydroelectric Reservoirs and Global Warming". In: **Proceedings of the RIO 02 - World Climate & Energy Event**, pp. 123-129, Rio de Janeiro, Jan. 2002.

SAKELLARIOU, R., ZHAO, H., TSIAKKOURI, E., *et al.* "Scheduling Workflows with Budget Constraints". In: **Proceedings of the CoreGRID Integration Workshop 2005**, pp. 189-202, Pisa, Italy, Nov. 2005.

SEMERARO, G., MAGKLIS, G., BALASUBRAMONIAN, R., *et al.* "Energy-efficient

processor design using multiple clock domains with dynamic voltage and frequency scaling". In: **Proceedings of the Eighth International Symposium on High-Performance Computer Architecture (HPCA 2002)**, pp. 29-42, Boston, Massachusettes, Fev. 2002.

SETI@HOME. *SETI@home Grid Project*. [S.l: s.n.]: [S.d.]. Disponível em: <http://setiathome.berkeley.edu/>. Acesso em: 21 nov. 2011.

SHARMA, S., HSU, C., FENG, W. "Making a case for a Green500 list". In: **Proceedings of the 20th IEEE International Parallel and Distributed Processing Symposium (IPDPS 2006)**, pp. 8, Rhodes Island, Greece, Abr. 2006.

SMITH, W., TAYLOR, V. E., FOSTER, I. T. "Using Run-Time Predictions to Estimate Queue Wait Times and Improve Scheduler Performance". In: **Proceedings of the Job Scheduling Strategies for Parallel Processing in conjunction with IPPS/SPDP '99**, pp. 202–219, San Juan, Puerto Rico, Abr. 1999.

SONMEZ, O., YIGITBASI, N., IOSUP, A., *et al.* "Trace-Based Evaluation of Job Runtime and Queue Wait Time Predictions in Grids". In: **Proceedings of the 18th ACM International ACM Symposium on High Performance Distributed Computing (HPDC 09)**, pp. 111–120, Munich, Germany, Jun. 2009.

SPECPOWER. *SPECpower_ssj2008*. [S.l: s.n.]: [S.d.]. Disponível em: <http://www.spec.org/power_ssj2008/>. Acesso em: 19 jan. 2012.

STEUER, R. E. **Multiple Criteria Optimization: Theory, Computation, and Application**. John Wiley, 1986.

STEVENS, R., ZHAO, J., GOBLE, C. "Using provenance to manage knowledge of In Silico experiments", **Briefings in Bioinformatics**, v. 8, n. 3, pp. 183 -194, Mai. 2007.

SULI, Y. "Benchmarking Computers for High Energy Physics Computing". In: **Proceedings of the Summer Student Programme**, Zeuthen, Germany, Ago. 2008.

TAYLOR, I., SHIELDS, M., WANG, I., *et al.* "Triana Applications within Grid Computing and Peer to Peer Environments", **Journal of Grid Computing**, v. 1, n. 2, pp. 199-217, 2003.

TESAURO, G., DAS, R., CHAN, H., *et al.* "Managing Power Consumption and Performance of Computing Systems Using Reinforcement Learning". In: **Proceedings of the 21st Annual Conference on Neural Information Processing Systems**, pp. 1-8, Vancouver, Canada, Dez. 2007.

THAIN, D., BENT, J., ARPACI-DUSSEAU, A., *et al.* "Gathering at the Well: Creating Communities for Grid I/O". In: **Proceedings of the 2001 ACM/IEEE conference on Supercomputing**, pp. 21-30, Denver, Colorado, Nov. 2001.

TOPCUOGLU, H., HARIRI, S., MIN-YOU WU. "Performance-effective and low-complexity task scheduling for heterogeneous computing", **IEEE Transactions on Parallel and Distributed Systems**, v. 13, n. 3, pp. 260-274, Mar. 2002.

TPC-ENERGY. *TPC-Energy*. [S.l: s.n.]: [S.d.]. Disponível em:

<http://www.tpc.org/tpc_energy/default.asp>. Acesso em: 19 jan. 2012.

TRIANA. *Triana Project*. [S.l: s.n.]: [S.d.]. Disponível em: <http://www.trianacode.org/>. Acesso em: 29 dez. 2011.

VERMA, A., AHUJA, P., NEOGI, A. "Power-aware Dynamic Placement of HPC Applications". In: **Proceedings of the 22nd Annual International Conference on Supercomputing (ICS 2008)**, pp. 175–184, Island of Kos, Greece, Jun. 2008.

VOLCKAERT, B., THYSEBAERT, P., TURCK, F. DE, *et al.* "Evaluation of Grid Scheduling Strategies Through a Network-Aware Grid Simulator". In: **Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA 2003)**, pp. 31-35, Las Vegas, USA, Jun. 2003.

WISSNER-GROSS, A. *Green Certified Site*. [S.l: s.n.]: [S.d.]. Disponível em: <http://www.co2stats.com/>. Acesso em: 1 nov. 2011.

WLCG. *Worldwide LHC Computing Grid*. [S.l: s.n.]: 2002. Disponível em: <http://lcg.web.cern.ch/lcg/>. Acesso em: 29 out. 2011.

WROE, C., GOBLE, C., GODERIS, A., *et al.* "Recycling workflows and services through discovery and reuse: Research Articles", **Concurrency and Computation Practice and Experience**, v. 19, n. 2, pp. 181–194, 2007.

YU, J., BUYYA, R. "A Novel Architecture for Realizing Grid Workflow using Tuple Spaces". In: **Proceedings of the 5th IEEE/ACM International Workshop on Grid Computing (GRID 2004)**, pp. 119–128, Pittsburgh, USA, Nov. 2004.

YU, J., BUYYA, R. "A Taxonomy of Workflow Management Systems for Grid Computing", **Journal of Grid Computing**, v. 3, n. 3-4, pp. 171-200, Set. 2005.

YU, J., BUYYA, R. "A Budget Constrained Scheduling of Workflow Applications on Utility Grids using Genetic Algorithms". In: **Proceedings of the Workshop on Workflows in Support of Large-Scale Science (WORKS 2006)**, pp. 1-10, Paris, France, Nov. 2006.

YU, J., BUYYA, R., RAMAMOHANARAO, K. "Workflow Scheduling Algorithms for Grid Computing". In: Xhafa, F., Abraham, A. (eds), **Metaheuristics for Scheduling in Distributed Computing Environments**, chapter 5, Heidelberg, Springer Berlin, 2008.

Apêndices

O Apêndice 1 apresenta os tempos de execução das estratégias HGreen e GGreen.

O Apêndice 2 contém um resumo do código fonte, em linguagem Java, da classe responsável pela implementação da estratégia HGreen, enquanto o Apêndice 3 apresenta a parte principal do código fonte de GGreen.

O Apêndice 4 exibe o código fonte, em linguagem C, do programa BOTEEN.

Apêndice 1 – Tempos de Execução de HGreen e GGreen

A seguir, as Tabelas 22, 23 e 24 apresentam os tempos de execução das estratégias HGreen e GGreen considerando os cenários C1, C2 e C3, respectivamente.

Tabela 22: Tempos de execução de HGreen e GGreen para C1 (em hora)

Lote	12 Tarefas		24 Tarefas		36 Tarefas		48 Tarefas	
	HGreen	GGreen	HGreen	GGreen	HGreen	GGreen	HGreen	GGreen
LT101	473,9	485,0	464,9	566,0	473,9	708,0	483,9	896,7
LT102	445,0	496,1	455,0	496,1	436,1	684,7	483,9	967,7
LT103	416,2	416,2	483,9	613,7	483,9	849,0	455,0	566,0
LT104	416,2	451,7	436,1	485,0	483,9	943,4	473,9	991,0
LT105	338,5	375,1	455,0	519,4	483,9	825,7	488,3	849,0
LT106	483,9	551,5	416,2	496,0	483,9	754,7	464,9	779,0
LT107	455,0	455,0	464,9	471,7	464,9	590,4	483,9	873,4
LT108	416,2	473,9	473,9	613,7	473,9	802,4	509,4	991,0
LT109	436,1	440,6	473,9	519,4	464,9	825,7	455,0	849,0
LT110	464,9	496,1	483,9	519,4	483,9	637,0	483,9	943,4
LT111	464,9	518,2	455,0	637,0	455,0	708,0	483,9	1014,4
LT112	436,1	462,8	406,2	590,4	455,0	731,4	473,9	802,4
LT113	329,5	364,0	483,9	542,7	483,9	754,7	473,9	802,4
LT114	426,1	426,1	483,9	779,0	445,0	731,4	531,5	943,4
LT115	464,9	496,1	455,0	590,4	483,9	754,7	488,3	1014,4
LT116	368,4	419,4	436,1	451,7	483,9	754,7	464,9	708,0
LT117	406,2	408,3	483,9	637,0	455,0	637,0	445,0	731,4
LT118	436,1	436,1	473,9	542,7	483,9	708,0	483,9	849,0
LT119	406,2	406,2	436,1	660,4	473,9	779,0	445,0	779,0
LT120	483,9	529,3	455,0	684,7	445,0	637,0	464,9	802,4
LT121	473,9	473,9	416,2	471,7	455,0	684,7	483,9	684,7
LT122	436,1	436,1	473,9	684,7	445,0	590,4	473,9	920,0
LT123	445,0	445,0	445,0	519,4	473,9	660,4	483,9	825,7
LT124	397,2	451,7	464,9	485,0	464,9	779,0	483,9	991,0
LT125	473,9	518,2	473,9	540,4	473,9	779,0	531,5	943,4
LT126	445,0	445,0	483,9	518,2	473,9	802,4	488,3	967,7
LT127	406,2	406,2	445,0	590,4	473,9	849,0	483,9	708,0
LT128	483,9	518,2	483,9	496,1	483,9	754,7	464,9	873,4
LT129	464,9	507,1	483,9	519,4	473,9	754,7	483,9	991,0
LT130	464,9	464,9	464,9	473,9	445,0	613,7	483,9	967,7
LT131	473,9	540,4	483,9	540,4	445,0	613,7	483,9	967,7
LT132	464,9	464,9	473,9	473,9	483,9	708,0	473,9	802,4
LT133	397,2	408,3	455,0	542,7	473,9	967,7	467,2	849,0
LT134	483,9	483,9	483,9	566,0	483,9	825,7	483,9	825,7
LT135	455,0	507,1	455,0	473,9	483,9	754,7	488,3	849,0
LT136	455,0	455,0	483,9	529,3	445,0	802,4	483,9	1014,4
LT137	464,9	464,9	473,9	485,0	426,1	684,7	464,9	731,4
LT138	397,2	440,6	455,0	496,0	483,9	802,4	483,9	849,0
LT139	483,9	551,5	464,9	542,7	483,9	754,7	473,9	943,4
LT140	436,1	485,0	406,2	519,4	455,0	708,0	473,9	849,0
LT141	483,9	483,9	397,2	613,7	483,9	731,4	483,9	920,0
LT142	464,9	507,1	464,9	518,2	445,0	708,0	464,9	802,4
LT143	483,9	483,9	483,9	496,1	473,9	896,7	464,9	873,4
LT144	445,0	451,7	464,9	473,9	473,9	637,0	473,9	802,4
LT145	406,2	451,7	473,9	779,0	483,9	779,0	464,9	802,4
LT146	455,0	455,0	483,9	551,5	445,0	637,0	483,9	873,4
LT147	261,8	76,3	397,2	448,4	483,9	684,7	464,9	849,0
LT148	483,9	483,9	483,9	542,7	483,9	873,4	473,9	920,0
LT149	426,1	426,1	483,9	660,4	473,9	731,4	483,9	825,7
LT150	455,0	455,0	455,0	613,7	464,9	873,4	473,9	637,0

Tabela 23: Tempos de execução de HGreen e GGreen para C2 (em hora)

Lote	12 Tarefas		24 Tarefas		36 Tarefas		48 Tarefas	
	HGreen	**GGreen**	**HGreen**	**GGreen**	**HGreen**	**GGreen**	**HGreen**	**GGreen**
LT101	483,9	551,5	436,1	462,8	464,9	496,1	483,9	540,4
LT102	406,2	419,4	329,5	329,5	483,9	483,9	473,9	529,3
LT103	377,3	377,3	483,9	485,0	455,0	496,1	483,9	731,4
LT104	483,9	483,9	300,7	341,8	483,9	529,3	464,9	496,0
LT105	368,4	368,4	483,9	483,9	483,9	529,3	483,9	518,2
LT106	455,0	507,1	426,1	462,8	483,9	551,5	464,9	873,4
LT107	377,3	377,3	387,3	387,3	377,3	397,2	473,9	529,3
LT108	455,0	455,0	473,9	518,2	464,9	485,0	455,0	519,4
LT109	270,8	286,3	473,9	485,0	406,2	406,2	397,2	425,0
LT110	348,5	348,5	455,0	455,0	473,9	529,3	483,9	518,2
LT111	445,0	445,0	445,0	445,0	473,9	473,9	483,9	660,4
LT112	464,9	464,9	329,5	352,9	397,2	429,5	464,9	529,3
LT113	406,2	406,2	464,9	507,1	445,0	496,1	473,9	613,7
LT114	261,8	265,2	483,9	551,5	483,9	519,4	464,9	566,0
LT115	251,9	251,9	464,9	464,9	416,2	451,7	473,9	754,7
LT116	348,5	348,5	329,5	329,5	483,9	483,9	455,0	507,1
LT117	426,1	485,0	455,0	455,0	455,0	507,1	445,0	471,7
LT118	483,9	529,3	464,9	529,3	455,0	471,7	473,9	660,4
LT119	416,2	416,2	464,9	507,1	436,1	451,7	483,9	551,5
LT120	416,2	416,2	416,2	416,2	473,9	485,0	473,9	731,4
LT121	358,4	408,3	483,9	496,1	483,9	529,3	436,1	566,0
LT122	309,6	309,6	416,2	440,6	455,0	613,7	455,0	825,7
LT123	436,1	436,1	483,9	529,3	436,1	436,1	436,1	451,7
LT124	416,2	416,2	455,0	455,0	483,9	483,9	436,1	473,9
LT125	387,3	387,3	455,0	455,0	464,9	542,7	473,9	507,1
LT126	358,4	386,2	416,2	416,2	464,9	507,1	483,9	802,4
LT127	309,6	341,8	406,2	406,2	464,9	566,0	445,0	496,1
LT128	464,9	518,2	319,6	319,6	455,0	507,1	483,9	551,5
LT129	233,0	265,2	483,9	518,2	445,0	496,1	473,9	590,4
LT130	416,2	429,5	483,9	551,5	483,9	518,2	483,9	590,4
LT131	416,2	416,2	473,9	507,1	483,9	483,9	473,9	873,4
LT132	436,1	436,1	483,9	483,9	483,9	507,1	455,0	507,1
LT133	473,9	473,9	329,5	329,5	445,0	485,0	483,9	483,9
LT134	387,3	387,3	397,2	429,5	473,9	473,9	473,9	684,7
LT135	464,9	496,1	455,0	455,0	436,1	436,1	473,9	754,7
LT136	483,9	551,5	397,2	419,4	483,9	485,0	455,0	684,7
LT137	329,5	330,7	473,9	473,9	368,4	419,4	473,9	473,9
LT138	397,2	429,5	445,0	451,7	464,9	464,9	483,9	540,4
LT139	377,3	377,3	436,1	496,1	473,9	566,0	483,9	483,9
LT140	233,0	233,0	455,0	507,1	397,2	397,2	426,1	496,0
LT141	416,2	473,9	483,9	496,1	416,2	416,2	473,9	507,1
LT142	436,1	436,1	483,9	483,9	483,9	507,1	483,9	551,5
LT143	426,1	426,1	464,9	464,9	436,1	436,1	483,9	519,4
LT144	483,9	507,1	464,9	464,9	455,0	462,8	483,9	613,7
LT145	464,9	464,9	473,9	540,4	464,9	529,3	464,9	613,7
LT146	426,1	426,1	464,9	496,1	473,9	529,3	455,0	519,4
LT147	483,9	507,1	406,2	451,7	473,9	540,4	416,2	473,9
LT148	416,2	462,8	406,2	451,7	483,9	708,0	455,0	455,0
LT149	483,9	483,9	426,1	451,7	473,9	540,4	455,0	590,4
LT150	445,0	445,0	436,1	485,0	445,0	496,1	483,9	542,7

Tabela 24: Tempos de execução de HGreen e GGreen para C3 (em hora)

Lote	12 Tarefas		24 Tarefas		36 Tarefas		48 Tarefas	
	HGreen	GGreen	HGreen	GGreen	HGreen	GGreen	HGreen	GGreen
LT101	158,6	385,0	876,8	974,4	24,3	24,4	731,4	731,4
LT102	114,3	278,5	953,3	1058,8	917,8	930,0	235,2	235,2
LT103	473,9	1156,4	978,9	1087,6	783,5	793,6	1130,9	1130,9
LT104	254,1	618,2	220,9	245,3	681,5	691,4	820,2	820,2
LT105	120,9	292,9	125,4	139,8	689,2	698,1	915,6	915,6
LT106	39,9	97,6	948,9	1054,3	1049,9	1064,3	263,0	263,0
LT107	247,4	601,5	788,0	875,6	330,7	335,1	1176,4	1176,4
LT108	483,9	1179,7	1017,6	1129,8	97,6	98,7	999,9	999,9
LT109	382,8	932,2	236,3	261,9	486,1	492,7	937,8	937,8
LT110	71,0	173,0	962,2	1068,7	173,0	175,3	1176,4	1176,4
LT111	34,4	83,2	223,1	248,5	816,8	827,9	43,3	43,3
LT112	203,0	493,8	490,5	544,9	653,7	662,5	660,3	660,3
LT113	194,1	471,7	524,9	582,7	943,4	956,6	430,6	430,6
LT114	188,6	458,3	699,1	775,8	756,9	768,0	890,0	890,0
LT115	483,9	1179,7	632,6	703,6	146,5	148,7	896,7	896,7
LT116	338,4	823,5	369,6	410,6	573,7	581,5	113,1	113,1
LT117	109,8	267,4	146,5	163,1	988,8	1002,1	958,9	958,9
LT118	406,2	991,0	567,1	630,3	363,9	368,4	17,7	17,7
LT119	386,2	941,1	597,0	662,6	363,9	368,4	875,6	875,6
LT120	76,6	186,4	236,3	261,9	264,1	268,5	279,7	279,7
LT121	347,4	846,8	318,5	354,0	173,0	175,3	590,4	590,4
LT122	283,0	689,2	162,0	179,7	797,9	808,0	457,2	457,2
LT123	336,2	819,0	820,1	911,1	523,8	531,5	598,2	598,2
LT124	136,5	332,9	486,1	540,5	590,4	598,2	980,0	980,0
LT125	66,5	160,9	941,1	1045,4	806,8	817,9	53,2	53,2
LT126	121,0	295,2	624,8	693,6	622,6	631,5	710,3	710,3
LT127	400,6	976,6	365,1	406,1	288,5	291,9	1045,4	1045,4
LT128	145,4	354,0	1041,0	1156,4	1092,1	1107,5	476,1	476,1
LT129	144,3	351,8	925,6	1028,8	311,8	316,2	839,0	839,0
LT130	254,1	618,2	864,5	960,0	672,5	681,4	325,2	325,2
LT131	35,4	85,4	390,7	434,0	450,6	457,2	1039,9	1039,9
LT132	402,8	981,1	420,6	467,2	366,2	370,7	571,5	571,5
LT133	445,0	1085,4	437,3	486,1	255,2	258,6	492,7	492,7
LT134	432,8	1054,3	471,6	523,8	613,7	621,5	997,7	997,7
LT135	199,7	486,1	155,3	173,0	523,8	531,5	865,7	865,7
LT136	247,4	601,5	896,7	995,5	936,7	948,9	774,7	774,7
LT137	25,5	62,1	946,7	1052,1	1085,4	1099,8	966,6	966,6
LT138	352,9	861,2	667,0	741,3	861,2	873,4	110,9	110,9
LT139	473,8	1153,1	22,1	24,3	311,8	316,2	308,5	308,5
LT140	403,9	983,3	41,0	45,4	356,2	361,7	1021,0	1021,0
LT141	168,7	410,6	664,8	738,1	668,1	677,0	459,4	459,4
LT142	78,8	191,9	157,5	175,3	194,1	196,4	59,9	59,9
LT143	438,4	1068,7	611,5	679,2	689,2	698,1	698,1	698,1
LT144	145,4	354,0	890,0	988,8	1108,7	1124,2	660,3	660,3
LT145	85,4	207,5	535,0	594,8	913,3	925,6	540,5	540,5
LT146	305,2	743,5	150,9	167,6	186,4	189,7	811,2	811,2
LT147	369,6	901,2	195,3	217,5	1099,8	1114,3	1118,7	1118,7
LT148	334,0	813,5	29,9	33,3	844,6	855,7	349,5	349,5
LT149	387,3	943,4	1062,1	1179,7	1144,2	1159,8	540,5	540,5
LT150	463,8	1129,8	985,5	1094,3	997,7	1011,1	366,2	366,2

Apêndice 2 – Código Fonte de HGreen

```
/**
 * This class represents a case study where tasks are distributed considering
 * HGreen heuristics.
 *
 * Author: Fabio Coutinho
 * Date: August 2011
 */
public class HGreenCaseStudies extends GridSim {

    private Integer userId;
    private String name;
    private GridletList taskList;
    private GridletList executedTaskList;
    private int numberOfResources;
    private int scenarioID;
    private int taskSize;
    private final int typeOfScenario;
    private Hashtable<Integer, Double> energyConsumption;
    private double maxExecutionTime;

          public  HGreenCaseStudies(String  name,  int  numberOfResources,  int  typeOfScenario,  int  scenarioID,  int
taskSize) throws Exception {
                    super(name, 9600);
                    this.name = name;
               this.numberOfResources = numberOfResources;
               this.executedTaskList = new GridletList();
        // Gets an ID for this entity
        this.userId = new Integer( getEntityId(name) );
        System.out.println("Creating a grid user entity with name = " +
                name + ", and id = " + this.userId);

        // Creates a list of Gridlets (tasks) for this grid user according to the scenario
        this.taskSize = taskSize;
        this.typeOfScenario = typeOfScenario;
        switch (this.typeOfScenario) {
                    case 1:
                              this.taskList = createGridletToC1( this.userId );
                              break;

                    case 2:
                              this.taskList = createGridletToC2( this.userId );
                              break;
                    case 3:
                              this.taskList = createGridletToC3( this.userId );
                              break;
                    default:
                              break;
                    }

        this.scenarioID = scenarioID;
        this.writeTaskWeights();
        this.energyConsumption = new Hashtable<Integer, Double>();
        this.maxExecutionTime = 0.0;
          }

          public static MachineList createMachines(int numWNs, int numCores, int mipsRating) {
                    MachineList wnList =  new MachineList();
                    Machine m =  null;
                    for (int i = 0; i < numWNs; i++) {
                              m =  new Machine(i, numCores, mipsRating);
                              wnList.add(m);
                    }
                    return wnList;
          }

          public static GreenGridResource createGridResource(String name, String arch, int numCores,
                                                              int mipsRating, int eee, int powerLevel0,
                                                              int   powerLevel1,   int   powerLevel2,   int   powerLevel3,
                                                              int powerLevel4, int powerLevel5, int powerLevel6,
                                        int powerLevel7, int powerLevel8, int powerLevel9, int powerLevel10) {
                    System.out.println("Creating Grid Resources");
                    LinkedList<Integer> weekendList = new LinkedList<Integer>();
                    LinkedList<Integer> holidayList = new LinkedList<Integer>();
          ResourceCalendar resCal = new ResourceCalendar(1.0, 1, 1, 1, weekendList, holidayList, 1);

          // creating site s1
        MachineList mList = createMachines(1, numCores, mipsRating);

        // TIME_SHARED(round-robin scheduling) and SPACE_SHARED(queuing systems)
        ResourceCharacteristics resConfig = new ResourceCharacteristics(arch, "SL",
                    mList, ResourceCharacteristics.SPACE_SHARED, -3, 1);

        GreenGridResource gridResource = null;

        try
        {
          gridResource = new GreenGridResource(name, Link.DEFAULT_BAUD_RATE, resConfig, resCal, null, eee,
                                        powerLevel0, powerLevel1, powerLevel2, powerLevel3,
                                        powerLevel4, powerLevel5, powerLevel6, powerLevel7,
                                        powerLevel8, powerLevel9, powerLevel10);
        }
        catch (Exception e) {
            e.printStackTrace();
        }
                    return gridResource;
          }
```

```
          public static void createGridEnvironment() {

                    System.out.println("Creating Grid Resources");

        createGridResource("s1", "Supermicro 6025B", 8, 1174, 435, 219, 234, 248, 260, 273, 286, 298, 309, 318, 327,
334);
        createGridResource("s2", "HP ProLiant DL380 (Xeon 5160)",  4, 1535, 382, 172, 177, 182, 187, 195, 205, 218,
229, 242, 252, 258);
        createGridResource("s3", "HP ProLiant DL360 (Xeon X5570)", 8, 2548, 1586, 88, 117, 131, 144, 155, 167, 183,
197, 212, 230, 245);
        createGridResource("s4", "Dell 2970 (Opteron 2356)", 8, 1190, 545, 139, 164, 185, 205, 223, 236, 249, 263,
277, 292, 302);
        createGridResource("s5", "Bull SAS R440 (Xeon X5670)", 12, 2902, 2831, 60, 107, 122, 133, 144, 157, 173, 191,
211, 229, 247);
        createGridResource("s6", "SGI Altix XE320 (Xeon E5420)", 8, 1322, 681, 155, 172, 185, 197, 206, 214, 221, 231,
247, 255, 260);
          }

          public LinkedList<Integer> getSortedByEeeResourceIdList(LinkedList<Integer> resourceIdList) {
                    int maxEee = 0;
                    int resIDMaxSPECpower = -1;
                    int maxIndex = -1;

                    LinkedList<Integer> sortedByEeeResourceIdList = new LinkedList<Integer>();
                    LinkedList<Integer> tempResourceIdList = new LinkedList<Integer>();
                    tempResourceIdList.addAll(resourceIdList);

                    while (tempResourceIdList.size()>1)
                    {
                        for (int i = 0; i < tempResourceIdList.size(); i++)
                        {
                            // get Resource ID
                            int resourceID = ( (Integer)tempResourceIdList.get(i) ).intValue();

                            // Request to resource entity to send its SPECpower(eee)
                            super.send(resourceID, GridSimTags.SCHEDULE_NOW,
                                          GreenTags.RESOURCE_EEE, userId);

                            // waiting to get a resource eee
                            Integer objEee = (Integer) super.receiveEventObject();
                            int resEee = objEee.intValue();

                            if (resEee > maxEee)
                            {
                              maxEee = resEee;
                              maxIndex = i;
                              resIDMaxSPECpower = resourceID;
                            }
                        }
                        sortedByEeeResourceIdList.add(new Integer(resIDMaxSPECpower));
                        tempResourceIdList.remove(maxIndex);
          maxEee = -1;
                    }
          sortedByEeeResourceIdList.add(tempResourceIdList.getFirst());  // adds last resourceId (minimal eee)

                    return sortedByEeeResourceIdList;
          }

          public void allocTasksUpToFillResource(Integer resourceId)
          {
                    int resID = resourceId.intValue();
                    int indexPE=0;
                    double load = 0.00;
                    Gridlet execTask;

                    while ( (taskList.size() > 0) && (load < 100) )
                    {
                    super.send(resID, GridSimTags.SCHEDULE_NOW, GridSimTags.RESOURCE_CHARACTERISTICS, userId);

                              // waiting to get a resource characteristics
                              ResourceCharacteristics resChar = (ResourceCharacteristics) super.receiveEventObject();
                              if (resChar != null)
                              {
                                        load = (resChar.getNumPE() - resChar.getNumFreePE())*100 / resChar.getNumPE();
               // assigns the task to resource
                              Gridlet task = getMaxTask();
                              super.gridletSubmit(task, resID);
                        super.recordStatistics("\"Submitting task " + task.getGridletID() + " to " + resID + "\"", "");

                    execTask = super.gridletReceive(task.getGridletID(), userId, resID);
                    System.out.println("Receiving Gridlet " + execTask.getGridletID());

                    // Records this event into "stat.txt" file for statistical purposes
                        super.recordStatistics("\"Received gridlet " + execTask.getGridletID() +  " from resID " + resID +
"\"", "");
                              executedTaskList.add(execTask);
                              super.recordStatistics("==========  Execution  Time:  "  +  execTask.getActualCPUTime(),
"==========");

                              resChar.setStatusPE(PE.BUSY, 0, indexPE);
                    indexPE++;

                              taskList.remove(task);
                    if (resChar.getNumFreePE() == 0)
                        load = 100.0;
                              }
                        }
          }
```

```
/**
 * Creates the tasks (Gridlet objects) of the simulated workflow of scenario C1 (high variance)
 */
private GridletList createGridletToC1(Integer userID)
{
    long file_size = 300;
    long output_size = 300;
    // Creates a container to store Gridlets
    GridletList list = new GridletList();
    int limit = this.taskSize/2;

    // creating the first group of tasks (lightweight)
    int min = 15;
    int max = 50;
    long randomNumber;
    // first part of tasks with weight between 150000 and 500000 MI
    for (int i = 1; i <= limit; i++) {
      randomNumber = (long)(Math.random()*(max - min + 1)) + min;
      long weight = randomNumber*10000;
      Gridlet gridlet =  new Gridlet(i, weight, file_size, output_size);
      gridlet.setUserID(userID);
      list.add(gridlet);
                }
    // creating the second group of tasks (heavyweight)
    // last part of tasks with weight between 150 and 500 million MI
    for (int i = limit+1; i <= limit*2; i++) {
      randomNumber = (long)(Math.random()*(max - min + 1)) + min;
      long weight = randomNumber*10000000;
      Gridlet gridlet =  new Gridlet(i, weight, file_size, output_size);
      gridlet.setUserID(userID);
      list.add(gridlet);
                }
    System.out.println("Creating " + list.size() + " Tasks");
    return list;
}

/**
 * Creates the tasks (Gridlet objects) of the simulated workflow of scenario C2 (low variance)
 */
private GridletList createGridletToC2(Integer userID)
{
    long file_size = 300;
    long output_size = 300;
    // Creates a container to store tasks
    GridletList list = new GridletList();
    int limit = this.taskSize/3;
    // creating the first group of tasks (lightweight)
    int min = 5;
    int max = 50;
    long randomNumber;
    for (int i = 1; i <= limit; i++) {
      randomNumber = (long)(Math.random()*(max - min + 1)) + min;
      long weight = randomNumber*100000;
      Gridlet gridlet =  new Gridlet(i,  weight, file_size, output_size);
      gridlet.setUserID(userID);
      list.add(gridlet);
                }
    // creating the second group of tasks (middle-weight)
    for (int i = limit+1; i <= limit*2; i++) {
      randomNumber = (long)(Math.random()*(max - min + 1)) + min;
      long weight = randomNumber*1000000;
      Gridlet gridlet =  new Gridlet(i,  weight, file_size, output_size);
      gridlet.setUserID(userID);
      list.add(gridlet);
                }
    // creating the third group of tasks (heavy weight)
    for (int i = (limit*2)+1; i <= limit*3; i++) {
      randomNumber = (long)(Math.random()*(max - min + 1)) + min;
      long weight = randomNumber*10000000;
      Gridlet gridlet =  new Gridlet(i,  weight, file_size, output_size);
      gridlet.setUserID(userID);
      list.add(gridlet);
                }
    System.out.println("Creating " + list.size() + " Tasks");
    return list;
}

/**
 * Creates the tasks (Gridlet objects) of the simulated workflow of scenario C3 (same task weight)
 */
private GridletList createGridletToC3(Integer userID)
{
    long file_size = 300;
    long output_size = 300;
    // Creates a container to store Gridlets
    GridletList list = new GridletList();

    int min = 5;
    int max = 500;
    // creating tasks with weight between 5 and 500 milhoes de MI
    long randomNumber;
    randomNumber = (long)(Math.random()*(max - min + 1)) + min;
    long weight = randomNumber*1000000;
    for (int i = 1; i <= this.taskSize; i++) {
      Gridlet gridlet =  new Gridlet(i, weight, file_size, output_size);
      gridlet.setUserID(userID);
      list.add(gridlet);
                }

    System.out.println("Creating " + list.size() + " Tasks");
    return list;
}
```

```
public boolean taskWasExecuted(Gridlet task)
          {
                    for (int i = 0; i < executedTaskList.size(); i++) {
                              Gridlet t = (Gridlet)executedTaskList.get(0);
                              if (task.getGridletID() == t.getGridletID())
                                        return true;
                    }
                    return false;
          }
          public Gridlet getMaxTask() {
                    double maxLength = 0;
                    Gridlet maxGridlet = null;
                    for (int i = 0; i < this.taskList.size(); i++)
                    {
                              Gridlet gridlet = (Gridlet) this.taskList.get(i);
                              if ( gridlet.getGridletLength() > maxLength )
                              {
                                        maxLength = gridlet.getGridletLength();
                                        maxGridlet = gridlet;
                              }
                    }
                    return maxGridlet;
          }

    /**
     * The core method that handles communications among GridSim entities
     */
    public void body()
    {
        LinkedList<Integer> resourceIdList;

        // waiting to get list of resources.
        while (true)
        {
            // need to pause for a while to wait GridResources finish registering to GIS
            super.gridSimHold(1.0);    // hold by 1 second

            resourceIdList = super.getGridResourceList();
            System.out.println("Number of grid resources: " + resourceIdList.size());
            if (resourceIdList.size() == this.numberOfResources)
                break;
            else
                System.out.println("Waiting to get list of resources from GIS...");
        }

        // get all the resources available sorted by spec power benchmark
        LinkedList<Integer> sortedByEeeResourceIdList = getSortedByEeeResourceIdList(resourceIdList);

        // printResourceIDList(sortedByEeeResourceIdList);

        // assign "heavy tasks" to energy-efficient resources
        for (int i = 0; i < sortedByEeeResourceIdList.size(); i++)
                    allocTasksUpToFillResource(sortedByEeeResourceIdList.get(i));


        calculateMaxExecutionTime();

        calculateEnergyConsumption();

        printGridletList();

        printResourceCosts();

        // shut down all the entities, including GridStatistics entity since
        super.shutdownGridStatisticsEntity();
        super.shutdownUserEntity();
        super.terminateIOEntities();
    }

          public static void main(String[] args) {
                    try
                    {
                    int scenarioID_ = Integer.parseInt(args[0]);
                    int typeOfScenario_ = Integer.parseInt(args[1]);
                    int taskSize_ = Integer.parseInt(args[2]);

        System.out.println("Initializing HGreen GridSim Simulation, scenario " + scenarioID_);

        GridSim.init(1, Calendar.getInstance(), true, null, null, "report");

        GreenGridInformationService greenGIS = new GreenGridInformationService(GreenTags.GIS_NAME);

        GridSim.setGIS(greenGIS);

        createGridEnvironment();

        HGreenCaseStudies   cs1   =   new   HGreenCaseStudies("HGreenCaseStudies",   6,   typeOfScenario_,   scenarioID_,
taskSize_);
                  GridSim.startGridSimulation();
                    }
                    catch (Exception e)
        {
            e.printStackTrace();
            System.out.println("Unwanted errors happen");
        }
          }
```

```
/**
 * Prints the ResourceIDs and your position in list
 * @param resourceIDlist  list of resource IDs
 */
private void printResourceCosts()
{
      double energyConsumptionTotal = 0;
      for (int i = 0; i < super.getGridResourceList().size(); i++) {
                Integer resID = ((Integer)super.getGridResourceList().get(i));
                          System.out.print("Resource ID: " + resID);
                          Double wastedEnergy = (Double)this.energyConsumption.get(resID);
                          System.out.println(" Energy Consumption(Watt-hour): " + wastedEnergy/3600 + " Wh");
                          energyConsumptionTotal = energyConsumptionTotal + (wastedEnergy/3600);
                }
      System.out.println("Energy Consumption Total: " + energyConsumptionTotal);
}

      /**
 * Gets the tasks list executed in the resource identified by resID
 * @param resID resource identification
 */
private GridletList getTaskListExecutedByResource(int resID)
{
      GridletList resourceTasks = new GridletList();
      Gridlet gridlet;

      for (int i = 0; i < getSelectedGridletList().size(); i++) {
                gridlet = (Gridlet) getSelectedGridletList().get(i);
                if (gridlet.getResourceID() == resID)
                          resourceTasks.add(gridlet);
                }
      return resourceTasks;
}

/**
 * Gets the executed task with minimal Time Execution
 * @param list  list of tasks
 */
private Gridlet getTaskWithMinimalExecutionTime(LinkedList<Gridlet> list)
{
      double minExexTime = 999999999;
      Gridlet gridlet;
      Gridlet minGridlet = null;
      for (int i = 0; i < list.size(); i++) {
                gridlet = (Gridlet) list.get(i);
                if (gridlet.getActualCPUTime() < minExexTime) {
                          minExexTime = gridlet.getActualCPUTime();
                          minGridlet = gridlet;
                }
                }

      return minGridlet;
}

/**
 * Gets the cost to the load equals ZERO
 */
private double getCostToZeroLoad(int resID)
{
      super.send(resID, GridSimTags.SCHEDULE_NOW, GridSimTags.RESOURCE_CHARACTERISTICS, userId);

                // waiting to get a resource characteristics
                ResourceCharacteristics resChar = (ResourceCharacteristics) super.receiveEventObject();
      if (resChar != null)
                super.send(resID, GridSimTags.SCHEDULE_NOW, GreenTags.RESOURCE_POWER_LEVEL_0, userId);

      // waiting to get a resource characteristics
      Integer objLevel = (Integer) super.receiveEventObject();

      return objLevel.intValue();
}

      /**
 * Gets the cost according to the load of a resource
 * @param list  list of tasks
 */
private double getCostAccordingToLoad(LinkedList<Gridlet> taskList, int resID)
{
      //double load = 0.0;
      int load = 0;
                super.send(resID, GridSimTags.SCHEDULE_NOW, GridSimTags.RESOURCE_CHARACTERISTICS, userId);

                // waiting to get a resource characteristics
                ResourceCharacteristics resChar = (ResourceCharacteristics) super.receiveEventObject();
      if (resChar != null)
                load = (int) Math.round(taskList.size()/(double)resChar.getNumPE()*10); // obtém o nível de carga
                //load = taskList.size()*100 / resChar.getNumPE();

                switch (load) {
                          case 0:
                  // Request to resource entity to send its LEVEL_0
                  super.send(resID, GridSimTags.SCHEDULE_NOW,
                                       GreenTags.RESOURCE_POWER_LEVEL_0, userId);
                                    break;
                          case 1:
                                    // Request to resource entity to send its LEVEL_1
                  super.send(resID, GridSimTags.SCHEDULE_NOW,
                              GreenTags.RESOURCE_POWER_LEVEL_1, userId);
                  break;
                          case 2:
                  // Request to resource entity to send its LEVEL_2
                  super.send(resID, GridSimTags.SCHEDULE_NOW,
```

```
                                        GreenTags.RESOURCE_POWER_LEVEL_2, userId);
                        break;
                                case 3:
                        // Request to resource entity to send its LEVEL_3
                        super.send(resID, GridSimTags.SCHEDULE_NOW,
                                        GreenTags.RESOURCE_POWER_LEVEL_3, userId);
                        break;
                                case 4:
                        // Request to resource entity to send its LEVEL_4
                        super.send(resID, GridSimTags.SCHEDULE_NOW,
                                        GreenTags.RESOURCE_POWER_LEVEL_4, userId);
                                        break;
                                case 5:
                                        // Request to resource entity to send its LEVEL_5
                        super.send(resID, GridSimTags.SCHEDULE_NOW,
                                   GreenTags.RESOURCE_POWER_LEVEL_5, userId);
                        break;
                                case 6:
                        // Request to resource entity to send its LEVEL_6
                        super.send(resID, GridSimTags.SCHEDULE_NOW,
                                        GreenTags.RESOURCE_POWER_LEVEL_6, userId);
                        break;
                                case 7:
                        // Request to resource entity to send its LEVEL_7
                        super.send(resID, GridSimTags.SCHEDULE_NOW,
                                        GreenTags.RESOURCE_POWER_LEVEL_7, userId);
                        break;
                                case 8:
                        // Request to resource entity to send its LEVEL_8
                        super.send(resID, GridSimTags.SCHEDULE_NOW,
                                        GreenTags.RESOURCE_POWER_LEVEL_8, userId);
                        break;
                                case 9:
                        // Request to resource entity to send its LEVEL_9
                        super.send(resID, GridSimTags.SCHEDULE_NOW,
                                        GreenTags.RESOURCE_POWER_LEVEL_9, userId);
                        break;
                                default:
                        // Request to resource entity to send its LEVEL_10
                        super.send(resID, GridSimTags.SCHEDULE_NOW,
                                        GreenTags.RESOURCE_POWER_LEVEL_10, userId);
                                        break;
                }

                // waiting to get a resource characteristics
                Integer objLevel = (Integer) super.receiveEventObject();

        return objLevel.intValue();
    }

    /**
     * Calculates the Maximal Execution Time over all tasks.
     */
    private void calculateMaxExecutionTime()
    {
          Gridlet gridlet;
          double maxTimeExecution = 0;
          for (int i = 0; i < this.getSelectedGridletList().size(); i++) {
                    gridlet = (Gridlet) this.getSelectedGridletList().get(i);
                              if (gridlet.getActualCPUTime(gridlet.getResourceID()) > maxTimeExecution)
                                        maxTimeExecution = gridlet.getActualCPUTime(gridlet.getResourceID());
                    }
          this.maxExecutionTime = maxTimeExecution;
          System.out.println("Maximal Task Execution Time: " + this.maxExecutionTime);
    }


    /**
     * Calculates the Energy Consumption of the Resources and store them in energyConsumption Hashtable
     */
    private void calculateEnergyConsumption()
    {
          for (int i = 0; i < super.getGridResourceList().size(); i++) {
                    double resourceEnergyConsumption = 0.0;
                    double lastExecutionTime = 0;
                    int resourceId = ((Integer)super.getGridResourceList().get(i)).intValue();
                    GridletList resourceTaskList = getTaskListExecutedByResource(resourceId);

                    while (!resourceTaskList.isEmpty()) {
                    Gridlet minTask = getTaskWithMinimalExecutionTime(resourceTaskList);
                    double cost = getCostAccordingToLoad(resourceTaskList, resourceId);
                    resourceEnergyConsumption = resourceEnergyConsumption + cost*(minTask.getActualCPUTime(resourceId)
- lastExecutionTime);
                    System.out.print(">> TASK ID: " + minTask.getGridletID());
                    System.out.print(" | COST: " + cost);
                    System.out.print(" | TIME: " + (minTask.getActualCPUTime(resourceId) - lastExecutionTime) + "
seconds");
                    System.out.println(" | Partial Energy Cost: " + resourceEnergyConsumption);

                    lastExecutionTime = minTask.getActualCPUTime(resourceId);
                    resourceTaskList.remove(minTask);
                    }
                    // calculate complement time
                    double complementTime = this.maxExecutionTime - lastExecutionTime;
                    System.out.println("Complement Time: " + complementTime);
                    // complement time is the waiting time for heaviest task finishing in another resource
                    resourceEnergyConsumption              =              resourceEnergyConsumption              +
getCostToZeroLoad(resourceId)*complementTime;

                    energyConsumption.put(new Integer(resourceId), new Double(resourceEnergyConsumption));
                    }
    }
```

```
    /**
     * Writes the task weights to the output file
     */
          private void writeTaskWeights() {
                    try {
                              String strDirectory = "output/C" + this.typeOfScenario + "/" + this.taskSize;
                              new File(strDirectory).mkdirs();
                    File file = new File(strDirectory + "/tasks_" + this.scenarioID + ".txt");
                    if(file.exists()){
                              file.delete();
                    }
                    file.createNewFile();
                    //true = append file
                        FileWriter fileWritter = new FileWriter(file.getAbsolutePath(), true);
                  BufferedWriter bufferWritter = new BufferedWriter(fileWritter);

                  bufferWritter.write("Task_ID" + "\t" + "Weight");
                  for (int i = 0; i < this.taskList.size(); i++) {
                    Gridlet task = (Gridlet)taskList.get(i);
                    bufferWritter.write("\n" + task.getGridletID() + "\t" + (long)task.getGridletLength());
                              }
                  bufferWritter.close();

          } catch(IOException e){
                    e.printStackTrace();
          }

          }


    /**
     * Writes(appending) the energy consumption total to output file
     * @param energyConsumptionTotal  the energy consumption total
     */
          private void writeOutput(double energyConsumptionTotal) {
                    try {
                    File file = new File("output/C" + this.typeOfScenario + "/" + this.taskSize + "/results_" +
this.name +
                                                                                                        "_C" + this.typeOfScenario + "_" +
this.getSelectedGridletList().size() + "tasks" + ".txt");
                    boolean isNewFile = false;
                    //if file doesnt exists, then create it
                    if(!file.exists()){
                              file.createNewFile();
                              isNewFile = true;
                    }

                    //true = append file
                        FileWriter fileWritter = new FileWriter(file.getAbsolutePath(), true);
                  BufferedWriter bufferWritter = new BufferedWriter(fileWritter);
                  if (isNewFile)
                    bufferWritter.write("Scenario_ID" + "\t" + "Energy" + "\t" +"Time");
                  bufferWritter.write("\n"  +  this.scenarioID  +  "\t"  +  (long)energyConsumptionTotal  +  "\t"  +
Math.round(this.maxExecutionTime));
                  bufferWritter.close();

          } catch(IOException e){
                    e.printStackTrace();
          }

          }
```

Apêndice 3 – Código Fonte de GGreen

```
/**
 * This class represents a case study where tasks are distributed considering
 * GGreen heuristics.
 *
 * Author: Fabio Coutinho
 * Date: August 2011
 */
public class GGreenCaseStudies extends GridSim {

    private Integer userId;
    private String name;
    private GridletList taskList;
    private GridletList executedTaskList;
    private int numberOfResources;
    private int scenarioID;
    private final int typeOfScenario;
    private Hashtable<Integer, Double> energyConsumption;
    private double maxExecutionTime;

          public GGreenCaseStudies(String name, int numberOfResources, int typeOfScenario, int scenarioID, String
tasksPath) throws Exception {
                    super(name, 9600);
                    this.name = name;
                this.numberOfResources = numberOfResources;
                this.executedTaskList = new GridletList();
        // Gets an ID for this entity
        this.userId = new Integer( getEntityId(name) );
        System.out.println("Creating a grid user entity with name = " +
                name + ", and id = " + this.userId);
        this.scenarioID = scenarioID;
        this.typeOfScenario = typeOfScenario;
        // Creates a list of Gridlets (tasks) from a file previously executed by HGreenCaseStudies
        this.taskList = createGridletFromFile(this.userId, tasksPath);
        this.energyConsumption = new Hashtable<Integer, Double>();
        this.maxExecutionTime = 0.0;

          }
          public static MachineList createMachines(int numWNs, int numCores, int mipsRating) {
                    MachineList wnList =  new MachineList();
                    Machine m =  null;
                    for (int i = 0; i < numWNs; i++) {
                              m =  new Machine(i, numCores, mipsRating);
                              wnList.add(m);
                    }
                    return wnList;
          }

          public static GreenGridResource createGridResource(String name, String arch, int numCores,
                                                              int mipsRating, int cpe, int powerLevel0,
                                                              int  powerLevel1,  int  powerLevel2,  int  powerLevel3,
                                                              int  powerLevel4,  int  powerLevel5,  int  powerLevel6,'
                                                    int powerLevel7, int powerLevel8, int powerLevel9,int powerLevel10){
                    System.out.println("Creating Grid Resources");
                    LinkedList<Integer> weekendList = new LinkedList<Integer>();
                    LinkedList<Integer> holidayList = new LinkedList<Integer>();
          ResourceCalendar resCal = new ResourceCalendar(1.0, 1, 1, 1, weekendList, holidayList, 1);

          // creating site s1
        MachineList mList = createMachines(1, numCores, mipsRating);

        // TIME_SHARED(round-robin scheduling) and SPACE_SHARED(queuing systems)
        ResourceCharacteristics resConfig = new ResourceCharacteristics(arch, "SL",
                    mList, ResourceCharacteristics.SPACE_SHARED, -3, 1);

        GreenGridResource gridResource = null;

        try
        {
          gridResource = new GreenGridResource(name, Link.DEFAULT_BAUD_RATE, resConfig, resCal, null, cpe,
                                        powerLevel0, powerLevel1, powerLevel2, powerLevel3,
                                        powerLevel4, powerLevel5, powerLevel6, powerLevel7,
                                                  powerLevel8, powerLevel9, powerLevel10);
        }
        catch (Exception e) {
            e.printStackTrace();
        }
                    return gridResource;
          }

          /**
           *  Configures the environment and creates grid resources
           */
          public static void createGridEnvironment() {
          System.out.println("Creating Grid Resources");
        createGridResource("s1", "Supermicro 6025B", 8, 1174, 435, 219, 234, 248, 260, 273, 286, 298, 309, 318, 327,
334);
        createGridResource("s2", "HP ProLiant DL380 (Xeon 5160)",  4, 1535, 382, 172, 177, 182, 187, 195, 205, 218,
229, 242, 252, 258);
        //createGridResource("s3", "HP ProLiant DL360 (Xeon L5530)", 8, 2000, 2013, 62, 117, 153, 186);
        createGridResource("s3", "HP ProLiant DL360 (Xeon X5570)", 8, 2548, 1586, 88, 117, 131, 144, 155, 167, 183,
197, 212, 230, 245);
        createGridResource("s4", "Dell 2970 (Opteron 2356)", 8, 1190, 545, 139, 164, 185, 205, 223, 236, 249, 263,
277, 292, 302);
        //createGridResource("s5", "Bull SAS R440 (Xeon X5670)", 12, 1244, 2831, 60, 133, 191, 247);
        createGridResource("s5", "Bull SAS R440 (Xeon X5670)", 12, 2902, 2831, 60, 107, 122, 133, 144, 157, 173, 191,
211, 229, 247);
        createGridResource("s6", "SGI Altix XE320 (Xeon E5420)", 8, 1322, 681, 155, 172, 185, 197, 206, 214, 221, 231,
247, 255, 260);     }
```

```
public Gridlet getMaxTask() {
          double maxLength = 0;
          Gridlet maxGridlet = null;
          for (int i = 0; i < this.taskList.size(); i++)
          {
                    Gridlet gridlet = (Gridlet) this.taskList.get(i);
                    if ( gridlet.getGridletLength() > maxLength )
                    {
                              maxLength = gridlet.getGridletLength();
                              maxGridlet = gridlet;
                    }
          }
          return maxGridlet;
}

/**
 *  Gets resource power at receiving one more job
 */
public int getPowerWithNextJob(int resourceId)
{

          super.send(resourceId, GridSimTags.SCHEDULE_NOW, GridSimTags.RESOURCE_CHARACTERISTICS, userId);

          // waiting to get a resource characteristics
          ResourceCharacteristics resChar = (ResourceCharacteristics) super.receiveEventObject();
          int loadLevel = -1;

          if ( (resChar != null) && (resChar.getNumFreePE()>0) )
          {
                    int OC = resChar.getNumPE() - resChar.getNumFreePE();
                    loadLevel = (int) Math.round((OC+1)/(double)resChar.getNumPE()*10);

                    switch (loadLevel) {
                              case 0:
                      // Request to resource entity to send its LEVEL_0
                    super.send(resourceId, GridSimTags.SCHEDULE_NOW, GreenTags.RESOURCE_POWER_LEVEL_0, userId);
                    break;
                              case 1:
                                        // Request to resource entity to send its LEVEL_1
                    super.send(resourceId, GridSimTags.SCHEDULE_NOW, GreenTags.RESOURCE_POWER_LEVEL_1, userId);
                    break;
                              case 2:
                         // Request to resource entity to send its LEVEL_2
                    super.send(resourceId, GridSimTags.SCHEDULE_NOW, GreenTags.RESOURCE_POWER_LEVEL_2, userId);
                         break;
                              case 3:
                         // Request to resource entity to send its LEVEL_3
                    super.send(resourceId, GridSimTags.SCHEDULE_NOW, GreenTags.RESOURCE_POWER_LEVEL_3, userId);

                         break;
                              case 4:
                         // Request to resource entity to send its LEVEL_4
                    super.send(resourceId, GridSimTags.SCHEDULE_NOW, GreenTags.RESOURCE_POWER_LEVEL_4, userId);

                                        break;
                              case 5:
                                        // Request to resource entity to send its LEVEL_5
                    super.send(resourceId, GridSimTags.SCHEDULE_NOW, GreenTags.RESOURCE_POWER_LEVEL_5, userId);

                         break;
                              case 6:
                         // Request to resource entity to send its LEVEL_6
                    super.send(resourceId, GridSimTags.SCHEDULE_NOW, GreenTags.RESOURCE_POWER_LEVEL_6, userId);
                         break;
                              case 7:
                         // Request to resource entity to send its LEVEL_7
                    super.send(resourceId, GridSimTags.SCHEDULE_NOW, GreenTags.RESOURCE_POWER_LEVEL_7, userId);

                         break;
                              case 8:
                         // Request to resource entity to send its LEVEL_8
                    super.send(resourceId, GridSimTags.SCHEDULE_NOW, GreenTags.RESOURCE_POWER_LEVEL_8, userId);
                         break;
                              case 9:
                         // Request to resource entity to send its LEVEL_9
                    super.send(resourceId, GridSimTags.SCHEDULE_NOW, GreenTags.RESOURCE_POWER_LEVEL_9, userId);

                         break;
                              default:
                         // Request to resource entity to send its LEVEL_10
                   super.send(resourceId, GridSimTags.SCHEDULE_NOW, GreenTags.RESOURCE_POWER_LEVEL_10, userId);
                         break;
                        }

                        // waiting to get a resource characteristic (power)
                        Integer objLevel = (Integer) super.receiveEventObject();

          return objLevel.intValue();
          }
          else  // resChar.getNumFreePE() == 0
                    return -1;
}




/**
```

```
          *  Returns id resource that executes next job with minimal power
          */
         public int getMinPowerResource(LinkedList<Integer> resourceIdList)
         {

                  int minPower = 999999;
                  int minResId = -1;
                  for (int i = 0; i < resourceIdList.size(); i++)
             {
                 // get Resource ID
                 int resourceId = ( (Integer)resourceIdList.get(i) ).intValue();
                 int tempPower = getPowerWithNextJob(resourceId);
                 if ( (tempPower < minPower) && (tempPower != -1) )
                 {
                          minPower = tempPower;
                          minResId = resourceId;
                 }
                 else if (tempPower == minPower) { // in case of tie
                   if (resourceId < minResId) // minor Id is the tie-breaker
                            minResId = resourceId;
                 }
             }

                  return minResId;
         }

         public void allocNextTaskToMinPowerResource(Gridlet nextTask, int resourceId)
         {
                  Gridlet execTask;

                  super.send(resourceId, GridSimTags.SCHEDULE_NOW, GridSimTags.RESOURCE_CHARACTERISTICS, userId);

                  // waiting to get a resource characteristics
                  ResourceCharacteristics resChar = (ResourceCharacteristics) super.receiveEventObject();
                  if ( (resChar != null) && (resChar.getNumFreePE() > 0) )
                  {
                  // assigns the next task to min power resource
                           super.gridletSubmit(nextTask, resourceId);
                           super.recordStatistics("\"Submitting  task  "  +  nextTask.getGridletID()  +  "  to  "  +
resourceId + "\"", "");
                           System.out.println(" ******  Task " + nextTask.getGridletID() + " allocated to Resource
" + resChar.getResourceName() + "  ******");

            execTask = super.gridletReceive(nextTask.getGridletID(), userId, resourceId);
            System.out.println("Receiving Gridlet " + execTask.getGridletID());

            // Records this event into "stat.txt" file for statistical purposes
                           super.recordStatistics("\"Received gridlet " + execTask.getGridletID() +  " from resID "
+ resourceId + "\"", "");

                           executedTaskList.add(execTask);
                           super.recordStatistics("==========  Execution  Time:  "  +  execTask.getActualCPUTime(),
"==========");
                           resChar.setStatusPE(PE.BUSY, 0, resChar.getNumBusyPE());
                  }
         }

         public boolean taskWasExecuted(Gridlet task)
         {
                  for (int i = 0; i < executedTaskList.size(); i++) {
                           Gridlet t = (Gridlet)executedTaskList.get(0);
                           if (task.getGridletID() == t.getGridletID())
                                    return true;
                  }
                  return false;
         }


    /**
     * The core method that handles communications among GridSim entities
     */
    public void body()
    {
        LinkedList<Integer> resourceIdList;

        // waiting to get list of resources.
        while (true)
        {
            // need to pause for a while to wait GridResources finish registering to GIS
            super.gridSimHold(1.0);    // hold by 1 second

            resourceIdList = super.getGridResourceList();
            System.out.println("Number of grid resources: " + resourceIdList.size());
            if (resourceIdList.size() == this.numberOfResources)
                break;
            else
                System.out.println("Waiting to get list of resources from GIS...");
        }

        while (taskList.size() > 0)
                  {
            // get the resource operating with minimal power
            int idMinResource = getMinPowerResource(resourceIdList);
            if (idMinResource == -1) // all resources are fully busy
                    break;
            Gridlet nextTask = getMaxTask();
            allocNextTaskToMinPowerResource(nextTask, idMinResource);
            // removes the executed task
            taskList.remove(nextTask);
                  }

        calculateMaxExecutionTime();
```

```
        calculateEnergyConsumption();

        printGridletList();

        printResourceCosts();

        // shut down all the entities, including GridStatistics entity since
        super.shutdownGridStatisticsEntity();
        super.shutdownUserEntity();
        super.terminateIOEntities();
    }

          public static void main(String[] args) {
                    try
                    {
                              int scenarioID_ = Integer.parseInt(args[0]);
                              int typeOfScenario_ = Integer.parseInt(args[1]);
                              String tasksPath_ = args[2];

                  System.out.println("Initializing GGreen GridSim Simulation, scenario " + scenarioID_);

                  System.out.println("Tasks Path: " + tasksPath_);

                  GridSim.init(1, Calendar.getInstance(), true, null, null, "report");

                  GreenGridInformationService greenGIS = new GreenGridInformationService(GreenTags.GIS_NAME);

                  GridSim.setGIS(greenGIS);

                  createGridEnvironment();

                  GGreenCaseStudies cs1 = new GGreenCaseStudies("GGreenCaseStudies", 6, typeOfScenario_, scenarioID_,
tasksPath_);
                  GridSim.startGridSimulation();
                    }
                    catch (Exception e)
        {
            e.printStackTrace();
            System.out.println("Unwanted errors happen");
        }
          }

          /**
     * Prints the ResourceIDs and your position in list
     * @param resourceIDlist  list of resource IDs
     */
    private void printResourceCosts()
    {
          double energyConsumptionTotal = 0;
          for (int i = 0; i < super.getGridResourceList().size(); i++) {
                    Integer resID = ((Integer)super.getGridResourceList().get(i));
                              System.out.print("Resource ID: " + resID);
                              Double wastedEnergy = (Double)this.energyConsumption.get(resID);
                              System.out.println(" Energy Consumption(Watt-hour): " + wastedEnergy/3600 + " Wh");
                              energyConsumptionTotal = energyConsumptionTotal + (wastedEnergy/3600);
                    }
          System.out.println("Energy Consumption Total: " + energyConsumptionTotal);
    }

          /**
     * Gets the tasks list executed in the resource identified by resID
     * @param resID resource identification
     */
    private GridletList getTaskListExecutedByResource(int resID)
    {
          GridletList resourceTasks = new GridletList();
          Gridlet gridlet;

          for (int i = 0; i < getSelectedGridletList().size(); i++) {
                    gridlet = (Gridlet) getSelectedGridletList().get(i);
                    if (gridlet.getResourceID() == resID)
                              resourceTasks.add(gridlet);
                    }
          return resourceTasks;
    }

    /**
     * Gets the executed task with minimal Time Execution
     * @param list  list of tasks
     */
    private Gridlet getTaskWithMinimalExecutionTime(LinkedList<Gridlet> list)
    {
          double minExexTime = 999999999;
          Gridlet gridlet;
          Gridlet minGridlet = null;
          for (int i = 0; i < list.size(); i++) {
                    gridlet = (Gridlet) list.get(i);
                    if (gridlet.getActualCPUTime() < minExexTime) {
                              minExexTime = gridlet.getActualCPUTime();
                              minGridlet = gridlet;
                    }
                    }

          return minGridlet;
    }
```

```
     /**
     * Gets the cost to the load equals ZERO
     */
    private double getCostToZeroLoad(int resID)
    {
          super.send(resID, GridSimTags.SCHEDULE_NOW, GridSimTags.RESOURCE_CHARACTERISTICS, userId);

                    // waiting to get a resource characteristics
                    ResourceCharacteristics resChar = (ResourceCharacteristics) super.receiveEventObject();
          if (resChar != null)
                    super.send(resID, GridSimTags.SCHEDULE_NOW, GreenTags.RESOURCE_POWER_LEVEL_0, userId);

          // waiting to get a resource characteristics
                    Integer objLevel = (Integer) super.receiveEventObject();

          return objLevel.intValue();
    }

    /**
     * Gets the cost according to the load of a resource
     * @param list  list of tasks
     */
    private double getCostAccordingToLoad(LinkedList<Gridlet> taskList, int resID)
    {
          //double load = 0.0;
          int load = 0;
                    super.send(resID, GridSimTags.SCHEDULE_NOW, GridSimTags.RESOURCE_CHARACTERISTICS, userId);

                    // waiting to get a resource characteristics
                    ResourceCharacteristics resChar = (ResourceCharacteristics) super.receiveEventObject();
          if (resChar != null)
                    load = (int) Math.round(taskList.size()/(double)resChar.getNumPE()*10); // obtém o nível de carga
do recurso
                    //load = taskList.size()*100 / resChar.getNumPE();

                    switch (load) {
                              case 0:
                      // Request to resource entity to send its LEVEL_0
                      super.send(resID, GridSimTags.SCHEDULE_NOW,
                                        GreenTags.RESOURCE_POWER_LEVEL_0, userId);
                                  break;
                              case 1:
                                        // Request to resource entity to send its LEVEL_1
                      super.send(resID, GridSimTags.SCHEDULE_NOW,
                                GreenTags.RESOURCE_POWER_LEVEL_1, userId);
                      break;
                              case 2:
                      // Request to resource entity to send its LEVEL_2
                      super.send(resID, GridSimTags.SCHEDULE_NOW,
                                        GreenTags.RESOURCE_POWER_LEVEL_2, userId);
                      break;
                              case 3:
                      // Request to resource entity to send its LEVEL_3
                      super.send(resID, GridSimTags.SCHEDULE_NOW,
                                        GreenTags.RESOURCE_POWER_LEVEL_3, userId);
                      break;
                              case 4:
                      // Request to resource entity to send its LEVEL_4
                      super.send(resID, GridSimTags.SCHEDULE_NOW,
                                        GreenTags.RESOURCE_POWER_LEVEL_4, userId);
                                  break;
                              case 5:
                                        // Request to resource entity to send its LEVEL_5
                      super.send(resID, GridSimTags.SCHEDULE_NOW,
                                GreenTags.RESOURCE_POWER_LEVEL_5, userId);
                      break;
                              case 6:
                      // Request to resource entity to send its LEVEL_6
                      super.send(resID, GridSimTags.SCHEDULE_NOW,
                                        GreenTags.RESOURCE_POWER_LEVEL_6, userId);
                      break;
                              case 7:
                      // Request to resource entity to send its LEVEL_7
                      super.send(resID, GridSimTags.SCHEDULE_NOW,
                                        GreenTags.RESOURCE_POWER_LEVEL_7, userId);
                      break;
                              case 8:
                      // Request to resource entity to send its LEVEL_8
                      super.send(resID, GridSimTags.SCHEDULE_NOW,
                                        GreenTags.RESOURCE_POWER_LEVEL_8, userId);
                      break;
                              case 9:
                      // Request to resource entity to send its LEVEL_9
                      super.send(resID, GridSimTags.SCHEDULE_NOW,
                                        GreenTags.RESOURCE_POWER_LEVEL_9, userId);
                      break;
                              default:
                      // Request to resource entity to send its LEVEL_10
                      super.send(resID, GridSimTags.SCHEDULE_NOW,
                                        GreenTags.RESOURCE_POWER_LEVEL_10, userId);
                                  break;
                    }

                    // waiting to get a resource characteristics
                    Integer objLevel = (Integer) super.receiveEventObject();

          return objLevel.intValue();
    }
```

```
    /**
     * Process the line extracted from taskfile and return a task
     */
          private Gridlet processLine(String line, long file_size, long output_size) {
              //use a Scanner to parse the content of each line
              Scanner scanner = new Scanner(line);
              scanner.useDelimiter("\t");
              if (scanner.hasNext()) {
                      int id = scanner.nextInt();
                      long weight = scanner.nextLong();
                      return new Gridlet(id, weight, file_size, output_size);
              }
                    return null;
          }

    /**
     * Creates the tasks (Gridlet objects) of the simulated workflow from a file
     */
    private GridletList createGridletFromFile(Integer userID, String strFile)
    {
        long file_size = 300;
        long output_size = 300;
        // Creates a container to store Gridlets
        GridletList list = new GridletList();
        File file;
        BufferedReader inputReader = null;
                    try {
                              file = new File (strFile);
                              inputReader =  new BufferedReader(new FileReader(file));
                         String line = null;

                              while (( line = inputReader.readLine()) != null) {
                                        if (line.startsWith("Task_ID")) // descarta o cabecalho
                                                continue;
                                        Gridlet gridlet = processLine(line, file_size, output_size);
                                        if (gridlet != null)
                              gridlet.setUserID(userID);
                              list.add(gridlet);
                                 }
                    } catch (IOException ex){
                               ex.printStackTrace();
                    }

        System.out.println("Creating " + list.size() + " Tasks");
        return list;
    }

          /**
     * Writes(appending) the energy consumption total to output file
     * @param energyConsumptionTotal  the energy consumption total
     */
          private void writeOutput(double energyConsumptionTotal) {
                    try {
                    File     file     =     new     File("output/C"     +     this.typeOfScenario         +     "/"     +
this.getSelectedGridletList().size() + "/results_" + this.name +
                                                                                "_C"    +    this.typeOfScenario    +    "_"    +
this.getSelectedGridletList().size() + "tasks" + ".txt");
                    boolean isNewFile = false;
                    //if file doesnt exists, then create it
                    if(!file.exists()){
                              file.createNewFile();
                              isNewFile = true;
                    }

                    //true = append file
                        FileWriter fileWritter = new FileWriter(file.getAbsolutePath(), true);
                  BufferedWriter bufferWritter = new BufferedWriter(fileWritter);
                  if (isNewFile)
                    bufferWritter.write("Scenario_ID" + "\t" + "Energy" + "\t" +"Time");
                  bufferWritter.write("\n"  +  this.scenarioID  +  "\t"  +  (long)energyConsumptionTotal  +  "\t"  +
Math.round(this.maxExecutionTime));
                  bufferWritter.close();

          } catch(IOException e){
                    e.printStackTrace();
          }

          }
```

Apêndice 4 – Código Fonte de BOTEEN

```
#include <stdio.h>
#include <limits.h>

#define DATAFILE "C:\\Users\\fabio\\Documents\\C-Free\\Projects\\boteen\\entrada\\instancia.txt"
#define OUTPUTFILE "C:\\Users\\fabio\\Documents\\C-Free\\Projects\\boteen\\saida\\saida.txt"
#define OUTPUTFILE2 "C:\\Users\\fabio\\Documents\\C-Free\\Projects\\boteen\\saida\\saida - tempo e energia.txt"

#define N 200
#define M 24
/****DATA****/
long int e[N][M];  //colocar long int
int c[M];
long int t[N][M]; //colocar long int
int maq_ord[M];
int tar_ord[N];

int nPO = 0;

/*** OBTEM OS DADOS DE instancia.txt ***/
void carrega_dados()
{
  int i, j, k;
  //XPRBctr Obj, ctr;

  FILE *datafile;

  datafile=fopen(DATAFILE,"r");
  if(NULL==datafile)
    printf("ERRO - Arquivo de entrada");
  else
  {
    for(i=0;i<N;i++)
      for(j=0;j<M;j++)
        fscanf(datafile,"%ld ",&(e[i][j])); //%ld
    for(i=0;i<N;i++)
      for(j=0;j<M;j++)
        fscanf(datafile,"%ld ",&(t[i][j])); //%ld
    for(j=0;j<M;j++)
      fscanf(datafile,"%d ",&(c[j]));
  }
  fclose(datafile);
  printf("\n Fim carrega_dados \n");
}

//=============================================================================
void ordena()
{
          // gerando maq_ord
          int i,j;
          int temp_maq[M];
          int temp_tar[N];

          // copia vetor temporario
          for(j=0;j<M;j++)
              temp_maq[j] = e[0][j];

    // copia vetor temporario
          for(i=0;i<N;i++)
              temp_tar[i] = e[i][0];

          // ordenando maquinas
          int minJ;
          int jj;
          for(jj=0;jj<M;jj++)
          {
                    int minW = LONG_MAX;
                    for(j=0;j<M;j++)
                    {
                              if ( (temp_maq[j]!=-1) && (temp_maq[j] < minW) )
                              {
                                        minW=temp_maq[j];
                                        minJ = j;
                              }

                    }
                    maq_ord[jj]         = minJ;
                    temp_maq[minJ] = -1;
          }

          // ordenando tarefas
          int maxI;
          int ii;
          for(ii=0;ii<N;ii++)
          {
                    int maxW = -1;
                    for(i=0;i<N;i++)
                    {
                              if ( (temp_tar[i]!=-1) && (temp_tar[i] > maxW) )
                              {
                                        maxW=temp_maq[i];
                                        maxI = i;
                              }

                    }
                    tar_ord[ii]         = maxI;
                    temp_tar[maxI] = -1;
          }
}
```

```
/*** GRAVA RESULTADOS em saida.txt ***/
void grava_dados(long int **Xstar)
{
          FILE *outputfile;
          FILE *outputfile2;

          outputfile=fopen(OUTPUTFILE,"w");
          if(NULL==outputfile)
          {
                    printf("Erro ao abrir o arquivo de saida!\n");
                    exit(1);
          }

          outputfile2=fopen(OUTPUTFILE2,"w");
          if(NULL==outputfile2)
          {
                    printf("Erro ao abrir o arquivo de saida!\n");
                    exit(1);
          }

          fprintf(outputfile, "%s", "Numero de Solucoes: ");
          fprintf(outputfile, "%d\n", nPO);

          fprintf(outputfile2, "%s", "Numero de Solucoes: ");
          fprintf(outputfile2, "%d\n", nPO);
          int i, j;
          for(i=0; i<nPO; i++)
          {
                    for(j=0; j<N; j++)
                    {
                              fprintf(outputfile, "%s", "T_");
                              fprintf(outputfile, "%d", j+1);
                              fprintf(outputfile, "%s", " -> M_");
                              fprintf(outputfile, "%d\n", Xstar[i][j]+1);
                    }

                    fprintf(outputfile, "%s", "\nTempo = ");
                    fprintf(outputfile, "%ld", Xstar[i][N]);
                    fprintf(outputfile, "%s", " e Energia = ");
                    fprintf(outputfile, "%ld\n", Xstar[i][N+1]);

                    fprintf(outputfile2, "%ld\t", Xstar[i][N]);
                    fprintf(outputfile2, "%ld\n", Xstar[i][N+1]);
          }
          fclose(outputfile);
          printf("\n Fim grava_dados \n");
}

/*** USADO PARA AUXILIAR NA OBTENCAO DO VETOR DE TEMPOS DISTINTOS ***/
void quick_sort(int vetor[], int primeiro, int ultimo)
{
  int temp, baixo,alto,separador;
  baixo = primeiro;
  alto = ultimo;
  separador = vetor[(primeiro + ultimo) / 2];
  do
  {
    while(vetor[baixo] < separador)
      baixo++;
    while(vetor[alto] > separador)
      alto--;
    if(baixo <= alto)
    {
      temp =  vetor[baixo];
      vetor[baixo++] = vetor[alto];
      vetor[alto--] = temp;
    }
  }while(baixo <= alto);
  if(primeiro < alto)
    quick_sort(vetor,primeiro,alto);
  if(baixo < ultimo)
    quick_sort(vetor,baixo,ultimo);
}

//===========================================================================
/*** ENCONTRA k (QTDE DE TEMPOS DIST) E MONTA VET DE TMP DIST EM ORDEM CRES. ***/
int k_vet(int *v)
{
  int i, j, cont = 0;
  for(i=0;i<N;i++)
    for(j=0;j<M;j++)
      v[cont++] = t[i][j];

  quick_sort(v, 0, cont-1);
  int k = 1;
  for(i = 1; i < cont; i++)
    if(v[i] != v[i-1])
      k++;
  return k;
}

//===========================================================================
/*** T: TEMPOS DISTINTOS EM ORDEM DECRESCENTE ***/
void pesos_ordenados(int *T, int *v, int m)
{
  int i, pos = 1;
  T[0] = v[m-1];
  for(i=1; i<m; i++)
    if(v[m-i-1] != v[m-i])
      T[pos++] = v[m-i-1];
}
```

```
//=============================================================================
/*** OBTEM O TEMPO MAXIMO DE ALOC ***/
int tempo(int *Aloc)
{
          int i;
          long int tmp = -1;

          for(i=0;i<N;i++)
                    if (tmp < t[i][Aloc[i]])
                              tmp = t[i][Aloc[i]];

          return tmp;
}

//=============================================================================
/*** RETORNA O CONSUMO DE ENERGIA DA SOLUCAO y ***/
int energia(int *Aloc)
{
  int i;
  long int total_ener = 0;

  for(i=0;i<N;i++)
          total_ener = total_ener + e[i][Aloc[i]];

  return total_ener;
}

//=============================================================================
/*** DEFINE O SUBGRAFO G_u DADO PELO PARAMETRO u ***/
void armazena(long int **X, int *Aloc, long int tmp, long int ener)
{
          int i, j;

          for(i=0; i<N; i++)
                    X[nPO][i] = Aloc[i];

          X[nPO][N] = tmp;
          X[nPO][N+1] = ener;
          nPO++;
}

int MinEnergia(int *Aloc, long int trade_off)
{
          int k;
          int QtdeT[M];
          int i=0;
          int j=0;
          printf(" trade-off: ");
          printf(" %ld ", trade_off);
          // inicialmente, todas as maquinas estao vazias (sem tarefas)
          for(k=0; k<M; k++)
                    QtdeT[k]=0;

          while ( (i < N) && (j < M) )
          {
                    if ( (QtdeT[maq_ord[j]] < c[maq_ord[j]]) && (t[tar_ord[i]][maq_ord[j]] < trade_off) )
                    {
                              Aloc[tar_ord[i]]=maq_ord[j];
                              QtdeT[maq_ord[j]]++;
                              i++;
                              j = 0;
                    }
                    else
                              j++;
          }
          printf(" temp2.3 \n");
          if (j == M)
                    return 0;

          return 1;
}


int main(int argc, char *argv[])
{
          int i,j;

          carrega_dados();

          ordena();

          for(i=0; i<N; i++)
          {
                    for(j=0; j<M; j++)
                              printf("%ld  ", t[i][j]);
                    printf("\n");
          }
          printf("\n\n");

          for(i=0; i<N; i++)
          {
                    for(j=0; j<M; j++)
                              printf("%ld  ", e[i][j]);
                    printf("\n");
          }
          printf("\n\n");

          for(j=0; j<M; j++)
          {
                    printf("%d  ", c[j]);
          }
```

```
        long int **Xstar; // vai arm. o conj, min. completo de sol. pareto-otimas
        Xstar = (long int **)malloc((N*M) * sizeof(long int *));
        for(i=0; i<(N*M); i++)
                Xstar[i] = (long int *)malloc((N+2) * sizeof(long int));

        printf(" templ \n");

        int *Aloc1, *Aloc2;
        Aloc1 = (int *)malloc(N * sizeof(int));
        Aloc2 = (int *)malloc(N * sizeof(int));


        long int t_Aloc1, e_Aloc1, t_Aloc2, e_Aloc2;
        printf(" temp2 \n");
        MinEnergia(Aloc1, LONG_MAX);
        t_Aloc1 = tempo(Aloc1);
        e_Aloc1 = energia(Aloc1);

        while (1)
        {
                printf(" temp4 \n");
                if (!MinEnergia(Aloc2, t_Aloc1))
                {
                        printf(" temp5 \n");
                        armazena(Xstar, Aloc1, t_Aloc1, e_Aloc1);
                        break;
                }
                else
                {
                        printf(" temp6 \n");
                        t_Aloc2 = tempo(Aloc2);
                        e_Aloc2 = energia(Aloc2);
                        if (e_Aloc1 < e_Aloc2)
                                armazena(Xstar, Aloc1, t_Aloc1, e_Aloc1);
                        for(i=0;i<N;i++)
                                Aloc1[i]=Aloc2[i];
                        t_Aloc1 = t_Aloc2;
                        e_Aloc1 = e_Aloc2;
                }
        }

        grava_dados(Xstar);

        return 0;
}
```