Um estudo de caso sobre a avaliação energética de software como uma contribuição à Computação Verde

Jarbele C. da Silva[1], Alisson V. Brito[2], Gilberto F. de S. Filho[1]

[1]Centro de Ciências Aplicadas e Educação – Departamento de Ciência Exatas – Universidade Federal da Paraíba (UFPB) I – Campus IV – Paraíba – PB – Brasil

[2]Centro de Informática – Departamento de Informática – Universidade Federal da Paraíba (UFPB) – Campus I – Paraíba – PB – Brasil

{jarbele.cassia, gilberto}@dce.ufpb.br, alisson@ci.ufpb.br

Abstract. *To achieve the objectives proposed by the Green Computing it is necessary to consider techniques to reduce consumption, both on aspects of hardware and software. A major challenge is to develop methods that optimize components of all levels of abstraction. This includes the optimization of software components. In this context, this paper conducts a study dedicated to evaluating the energy expenditure of software applications through monitoring, analysis and testing, in order to examine how programming techniques may contribute to the development of more sustainable systems.*

Resumo. *Para alcançar os objetivos propostos pela Computação Verde é necessário considerar técnicas de redução de consumo, tanto em aspectos de hardware como de software. Um dos grandes desafios da Computação é desenvolver métodos de otimização de componentes em todos os níveis de abstração e isto inclui a redução de componentes de software. Neste contexto, este trabalho realiza um estudo dedicado à avaliação do dispêndio energético de aplicativos de software através de monitoramento, análise e testes, com o intuito de examinar como técnicas de programação podem contribuir com o desenvolvimento de sistemas mais sustentáveis.*

1. Introdução

Em paralelo aos processos de globalização têm-se acompanhado um gradativo crescimento na área de Tecnologia da Informação (TI), principalmente no que se refere ao desenvolvimento e consumo de dispositivos computacionais. A busca por aparelhos mais modernos, que ofereçam recursos inovadores e que satisfaçam os desejos de seus usuários tem acarretado um grande dispêndio tecnológico tanto no âmbito de hardware como de software. Prush (2011) aponta as principais tendências que tem direcionado esta concepção, bem como a gestão de TI, a citar: o aumento exponencial no número de dispositivos móveis; como corolário da Lei de Moore, a intensificação na densidade da potência dos dispositivos de TI; e o crescimento linear na densidade energética das baterias. Em contrapartida, essas tendências convergem no alto custo de energia para: chips de arquiteturas tradicionais, operação de centros de dados e a própria utilização da sociedade, afirma Prush (2011). Segundo ele, a vida útil da bateria tornou-se um fator limitante na utilidade de dispositivos móveis de TI.

Frente a estas novas realidades e a evolução dos sistemas computacionais têm-se diligenciado meios para promover a diminuição do consumo excessivo de energia

gerado pelos aparelhos eletrônicos. Fomentar a utilização dos recursos computacionais de forma consciente, proporcionando a preservação ambiental e a redução dos gastos na infraestrutura da TI [Rolt et. al. 2010] é característica chave para tentar ressarcir e regular esse consumo, assim como, lidar com o objetivo de ter uma Computação mais Verde.

Organizações passaram a realizar seus planejamentos e estratégias voltadas para um futuro mais sustentável, tendo como uma das principais bases a Tecnologia da Informação. Em 2006, a Sociedade Brasileira de Computação (SBC) publicou um relatório designando os cinco Grandes Desafios da Pesquisa em Computação no Brasil até o ano de 2016 [SBC 2006]. Os desafios propostos referem-se a:

- Gestão da informação em grandes volumes de dados multimídia distribuídos.
- Modelagem computacional de sistemas complexos artificiais, naturais e socioculturais e da interação homem-natureza.
- Impactos para a área da computação da transição do silício para as novas tecnologias.
- Acesso participativo e universal do cidadão brasileiro ao conhecimento.
- Desenvolvimento tecnológico de qualidade: sistemas disponíveis, corretos, seguros, escaláveis, persistentes e ubíquos.

Após seis anos, a comunidade cientifica brasileira caminha a passos largos, rumo a uma realidade compatível com esses objetivos [Mór et. al. 2010]. Entretanto, muitos avanços ainda precisam ser alcançados nos diferentes campos da Computação.

O trabalho apresentado neste artigo apoia-se no desafio "Impactos para a área da computação da transição do silício para novas tecnologias", considerando como alicerce a relação entre um dos principais fatores na construção de "sistemas verdes": o baixo consumo de energia elétrica, com a alta eficiência computacional [Mór et. al. 2010].

Para tanto, este trabalho tem como objetivo apontar, através da medição do consumo de energia de software, quais práticas de desenvolvimento de software devem ser adotadas na Computação Verde. Essa pesquisa demonstra que a medição no consumo de energia pode mostrar, de forma objetiva, quais sistemas são energeticamente mais eficientes e como utilizar recursos de hardware com menos custo energético.

Desta forma, o trabalho divide-se em seções descrevendo as ideias e resultados do estudo. A seção 2 apresenta uma contextualização sobre a Computação Verde listando as áreas da Computação que se dedicam à aplicação de estratégias para alcançar o objetivo da sustentabilidade computacional. Na seção 3 são descritas algumas ferramentas utilizadas para monitoramento do consumo de energia, dando uma maior ênfase na ferramenta Joulemeter, adotada em nossos experimentos. Na seção 4 são apresentados cenários para comparar o consumo energético de algumas técnicas de programação de aplicações de usuários. A seção 5 apresenta os resultados computacionais. E, a seção 6 é dedicada às considerações finais da pesquisa e a proposta de trabalhos futuros.

2. Computação Verde

A sociedade contemporânea tem usufruído fortemente de sistemas computacionais para efetuar diversas atividades, sejam elas nos lares, nas empresas, nas instituições de

ensino, nos ambientes comerciais etc. Em efeito da prevalência de tais tecnologias no cotidiano das pessoas é que o setor de TI (Tecnologia da Informação) tem sido um dos principais responsáveis pelo rápido desenvolvimento da sociedade e da economia, e consequentemente, contribuído para que o problema do consumo de energia na indústria de TI tenha se tornado cada vez mais agudo [Li e Zhou 2011]. Em virtude desses fatores e dos impactos irreversíveis que podem causar ao meio ambiente surgiu um novo paradigma capaz de contribuir com a necessidade de minimizar os custos e superar os problemas de consumo de energia causados pelo uso demasiado dos sistemas computacionais.

Dessa forma emerge o conceito de desenvolvimento sustentável. Na área da Computação, tal concepção é definida como Computação Sustentável, Computação Verde, ou ainda, como TI Verde, e compõe estratégias que visam reduzir o consumo de energia elétrica, bem como aumentar a eficiência de todos os processos e fenômenos relacionados à área [Albiero 2010].

2.1 Aplicação da Computação Verde

A Computação Verde (em inglês, Green Computing) é uma área da ciência que recentemente tem se dedicado ao estudo e a busca por alternativas que tornem mais econômicos o desempenho dos sistemas computacionais sem perder sua eficiência. Trata de uma nova forma de reconfigurar tais sistemas, de modo a disseminar a utilização mais eficiente da energia, recursos e insumos na produção de tecnologia, exigindo menos capacidade de processamento, e em consequência, promovendo a redução de custos e de consumo de energia.

Silva et al. (2010) aponta os focos principais aos quais esta área da ciência computacional se dedica: computação com o uso eficiente da energia; gerenciamento da energia; projetos de Data Centers verdes; virtualização de servidores; descarte responsável e reciclagem; utilização de fontes de energia renováveis e produtos de TI com selos ecológicos.

Uma pesquisa realizada no ano de 2008 pela Sun MicroSystems [Silva et. al. 2010 apud Murugesan 2008] apurou as causas fundamentais (Figura 1) para a utilização de práticas verdes, e constatou que a redução do consumo de energia elétrica (25%) e a redução de custos (24%), seguidos da redução de emissão de CO2 e do impacto ambiental (18%), da melhoria da performance dos sistemas (18%) e da redução de espaços (15%) são os principais benefícios gerados quando exercícios sustentáveis são postos em prática.

Figura 1. Causas fundamentais para a utilização de Práticas Sustentáveis

Para que estes resultados sejam concebidos estudos têm sido realizados na Computação e aplicados com o intuito de se obter sistemas computacionais mais sustentáveis. Entretanto, boa parte dos estudos com foco em TI Verde trata, apenas, da

medição e avaliação de desempenho energético de hardware propondo medidas de redução de consumo para tal.

Li e Zhou (2011) propõem três maneiras para medir e avaliar o consumo eficiente da energia, a citar: as técnicas da medição de hardware, a teoria computacional e a simulação de software. Segundo ele, o consumo de energia de aplicativos individuais reduz de modo mais eficaz, o calor produzido e a eletricidade consumida por tais aplicativos.

Mór et. al. (2010) demonstra que o paralelismo é um fator chave no processamento distribuído de grandes volumes de dados e que esse processo pode ser realizado com um consumo eficiente e escalável de energia.

Silva et. al. (2010) apresenta propostas para a implementação de procedimentos com vista a reduzir o consumo de energia por parte dos equipamentos de informática nas Instituições de Ensino Superior (IES), e apresenta as vantagens das soluções de computação em nuvem, *grid* computacional e virtualização.

Neto et. al. (2011) apresenta quatro técnicas para estimar e medir o consumo de energia da execução de aplicações em plataformas computacionais. As técnicas são: medição do nível de descarga da bateria para um notebook usando comandos do Linux; estimativa teórica com base nas características do processador, em um PC tipo desktop; simulação de uma arquitetura usando a ferramenta Sim-Panalyzer; e, medição do consumo real usando um osciloscópio.

Salgado (2011) investiga o consumo energético de máquinas virtuais e avalia a influência da redução da frequência de processamento no consumo energético de um *desktop*.

Os estudos já realizados geram avanços significativos na obtenção de práticas mais sustentáveis dentro deste contexto. Entretanto, boa parte das pesquisas realizadas com foco na redução do consumo de energia dos sistemas computacionais e eletrônicos tem sido voltada, especificamente, aos componentes de hardware. Os componentes de software têm sido menos considerados neste processo. Reis (2010) afirma que um projeto de sistemas computacionais visando a Computação Verde deve passar por um processo de redução de componentes, que vai desde a especificação do sistema - no maior nível de abstração possível - à sua síntese física.

Um dos grandes desafios da Computação Verde é desenvolver métodos de otimização de componentes em todos os níveis de abstração e isto inclui a redução de componentes de software, por exemplo: a criação de arquivos temporários, arquivos de apoio, entre outros métodos. Em seu estudo, Li e Zhou (2011) destacam a relevância de atribuir a responsabilidade do consumo de energia a aplicativos individuais, que podem também contribuir com a redução do calor produzido e a eletricidade consumida.

3. Ferramentas de Monitoração do Consumo de Energia

Os dispositivos de hardware, em geral, oferecem poucas ferramentas para medir o seu consumo energético, afirma Salgado (2011). Para isto, existem softwares específicos que realizam uma avaliação de consumo a partir de informações obtidas pelo sistema operacional de um dado dispositivo computacional.

3.1 Ferramentas de Monitoração

As ferramentas de monitoração ou medição são instrumentos que, através de escalas, gráficos ou dígitos, fornecem os valores numéricos das grandezas que estão sendo medidas [Albiero 2010]. Para realizar a monitoração de consumo de energia, neste trabalho, foram avaliadas diferentes ferramentas.

O HWMonitor é uma ferramenta gratuita que permite o monitoramento de hardware. Além de diagnosticar estes componentes também executa a verificação dos sensores térmicos presentes em cada um deles, informando sua respectiva temperatura, rotação dos coolers e voltagem [HWMonitor 2012]. Em sua interface, o HWMonitor expõe a voltagem e a temperatura na qual está operando no momento, além dos valores mínimos e máximos no qual pode operar. Por outro lado, a proposta deste aplicativo não está na verificação de componentes de software, fator que torna inviável a sua utilização neste trabalho.

Everest é um aplicativo utilizado na medição e registro do consumo e demanda de energia elétrica de um computador. Tal ferramenta monitora a temperatura dos dispositivos de hardware (processador, placa-mãe, cooler), oferece parâmetros para comparação e gera e exibe relatórios contendo os dados dos processos que estão sendo executados pelo computador [Everest 2007]. Entretanto, as informações de software exibidas pela ferramenta são muito técnicas e pouco precisas, o que dificulta a interpretação das mesmas. Trata-se de uma ferramenta eficaz para monitoração de componentes de hardware, e não exatamente de software.

Desenvolvido pela Microsoft para sistemas Windows, o Joulemeter [Joulemeter 2011] é uma ferramenta de medição e registro do consumo de energia elétrica, destinada a monitorar o dispêndio de máquinas virtuais (MVs), servidores, desktops, laptops e aplicativos de software. Por meio destas características ele é classificado como uma das melhores ferramentas que desempenham a medição e o monitoramento de consumo de energia em dispositivos computacionais.

PowerTop é uma ferramenta do GNU/Linux que provê informações sobre o consumo energético dos processos em execução [Salgado 2011]. Sua principal função é listar os processos que estão sendo executados no computador e verificar a porcentagem de processamento utilizado. Para avaliar a potência instantânea estimada pela máquina, o PowerTop utiliza o nível de carga da bateria. Desta forma é possível verificar os componentes de software que impedem o uso otimizado da energia em Sistemas Operacionais Linux.

A avaliação destas ferramentas foi feita com base nas condições relevantes para a realização do estudo. A Tabela 1 apresenta tais condições, bem como alguns dos critérios prioritários para seleção da mesma.

Tabela 1. Tabela avaliativa das ferramentas de monitoração analisadas

	Plataforma (Desktop)	SO (Windows)	Verifica Software	Fácil Manipulação	Registro de Dados
HWMonitor	x	X			X
Everest	x	X			X
Joulemeter	x	X	x	X	X
PowerTop	x		x		

A partir dessas estimativas percebeu-se que a ferramenta que mais se adequou as necessidades do estudo foi a ferramenta Joulemeter. A preferência por tal ferramenta justifica-se, principalmente, pela facilidade de manipulação e por permitir o monitoramento de componentes de software.

3.2. Ferramenta de monitoração Joulemeter

Além de monitorar o dispêndio energético de componentes de hardware, o Joulemeter também estima o consumo individual de dispositivos como CPU, memória, disco rígido e monitor. A ferramenta faz a estimativa através dos níveis de utilização dos componentes de hardware baseada em dados realísticos de consumo [Salgado 2011]. As avaliações são exibidas em watts (W) e registradas em um arquivo csv.

O Joulemeter enfoca dois aspectos relacionados com a otimização de energia: a modelagem de energia e a otimização de potência. Quanto à modelagem de energia: o Joulemeter estima o consumo de energia de uma máquina virtual, computador ou software através da medição dos recursos de hardware (CPU, disco, tela, memória, etc.) utilizados, convertendo o uso desses recursos para aplicação real de energia com base em modelos automáticos de cálculo de consumo [Joulemeter 2011]. Em seguida, é calculado o consumo total.

Quanto à otimização de potência, as funcionalidades fornecidas pela ferramenta podem ser utilizadas para otimizar o fornecimento de energia e os custos de consumo em vários cenários, que vão desde centros de dados à simples máquinas que funcionam com bateria. Por exemplo: a medição da potência da VM permite enxergar a capacidade de orçamento em data centers e desenvolver técnicas para a criação de data centers virtualizados; entre outros.

4. Estudo de caso

Vieira *et. al.* (2012) apresenta uma análise do desempenho e do consumo de energia de diferentes algoritmos por meio da utilização da Plataforma Android. Neste trabalho as análises feitas ocorreram com base em três momentos: análise de algoritmos recursivos e iterativos de mesma complexidade, análise de algoritmos que fazem mais uso de memória versus algoritmos que fazem mais uso de CPU e análise de alguns algoritmos de ordenação populares.

Fundamentados no estudo supracitado, esta pesquisa foi realizada seguindo um processo de análise constituído de três cenários de programação, sendo estes: o uso de dispositivos de entrada e saída, a complexidade de algoritmos, e o uso de recursos do sistema operacional.

Cenário 1: Uso de dispositivos de Entrada e Saída (E/S)

O primeiro cenário proposto testa o ganho energético no uso do disco rígido por uma aplicação. Para isso, duas estratégias de desenvolvimento são analisadas. Na primeira, o algoritmo possui uma camada de software que escreve diretamente na memória secundária cada mensagem recebida. A segunda estratégia adota uma camada de software contendo um *buffer*, na memória primária (memória RAM), logo, ao invés de escrever cada mensagem diretamente na memória secundária, as mensagens são armazenadas até preencher o buffer e escritas no dispositivo de uma vez.

Cenário 2: Complexidade de Algoritmos

No estudo da complexidade de algoritmo procura-se desenvolver e aplicar algoritmos mais eficientes na resolução de problemas. Neste cenário de teste, pretende-se avaliar a relação da complexidade de um algoritmo com seu custo energético.

Conforme Ascencio e Araujo (2011), para valores pequenos de *n* (dimensão da estrutura de dados de entrada), qualquer algoritmo, mesmo que ineficiente, gastará pouco tempo. O que faz o programador escolher um algoritmo não é o seu desempenho sobre tamanhos de entrada pequenos, mas sim sobre tamanhos de entrada grandes. Estuda-se então o comportamento assintótico da função de complexidade *T(n)*, onde a preocupação é a maneira como o tempo de execução de um algoritmo aumenta à medida que o tamanho da entrada aumenta.

Definição: uma função *g(n)* domina assintoticamente outra função *f(n)* se existem duas constantes positivas *c* e n_0 tais que, para $n \geq n_0$, temos que $|f(n)| \leq c \cdot |g(n)|$.

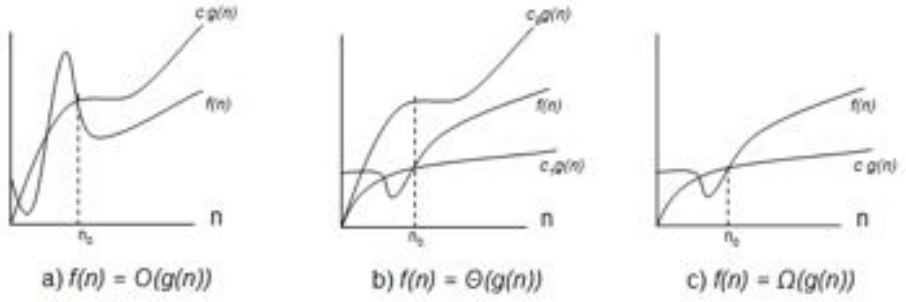

Figura 2. Exemplos gráficos das notações O, Ω e Θ.

Notação O : Uma função *f(n)* é *O(g(n))* se existem duas constantes positivas *c* e n_0 tais que $f(n) \leq c \cdot g(n)$, para todo $n \geq n_0$. A notação *O* é utilizada para dar um limite assintótico superior sobre uma função, dentro de um fator constante, a Figura 2a dá uma noção da notação *O*.

Notação Θ : Uma função *f(n)* é *Θ(g(n))* se existem constantes positivas c_1, c_2 e n_0 tais que $0 \leq c_1 \cdot g(n) \leq f(n) \leq c_2 \cdot g(n)$, para todo $n \geq n_0$. A notação *Θ* é utilizada para dar um limite assintótico firme sobre uma função, a Figura 2b dá uma noção da notação *Θ*.

Notação Ω : Uma função *f(n)* é *Ω(g(n))* se existem duas constantes positivas *c* e n_0 tais que $c \cdot g(n) \leq f(n)$, para todo $n \geq n_0$. A notação *Ω* é utilizada para dar um limite assintótico inferior sobre uma função, dentro de um fator constante, a Figura 2c dá uma noção da notação *Ω*.

Em nossa avaliação energética deste cenário são comparados quatro algoritmos clássicos de ordenação: Bubblesort, Selectionsort, Mergesort e Quicksort. É adotada a notação ***Θ*** de classificação dos algoritmos por ser a mais adequada à prática ao analisar os algoritmos em seu caso médio de eficiência.

Tabela 2. Avaliação assintótica *Θ* dos algoritmos de ordenação.

Algoritmo de ordenação	***Θ***
Bubblesort	n^2
Selectionsort	n^2
Mergesort	$n \log n$
Quicksort	$n \log n$

A Tabela 2 ilustra a avaliação assintótica Θ apresentada por Ascencio e Araujo (2011), para os algoritmos avaliados neste cenário.

Cenário 3: Algoritmo Iterativo X Algoritmo Recursivo

A recursividade é a propriedade que uma função tem de invocar a si próprio. Praticamente todas as linguagens de programação usadas atualmente permitem a especificação direta de funções e procedimentos recursivos. Sempre que uma função é invocada, a implementação da linguagem registra esta instância da função. Em muitas arquiteturas, para realizar este registro, usa-se uma pilha de chamada, embora outros métodos possam ser usados. Reciprocamente, toda função recursiva pode ser desenvolvida em uma função iterativa usando uma pilha.

Portanto, toda função que puder ser produzida por um computador pode ser escrita como função recursiva sem o uso de iteração; e qualquer função recursiva pode ser descrita através de iterações sucessivas.

Neste cenário pretende-se comparar o consumo de energia de dois algoritmos como mesma eficácia e eficiência, ou seja, que realizam a mesma tarefa e com a mesma avaliação assintótica de eficiência, entretanto implementados com técnicas distintas, sendo um algoritmo iterativo e outro recursivo.

Adotou-se neste cenário o algoritmo de percurso em pré-ordem sobre uma árvore binária, onde sua eficiência média é linear ($\Theta(n)$) tanto para a implementação iterativa quanto a recursiva, possibilitando assim apenas a comparação do consumo de energia das duas técnicas de programação.

5. Resultados Computacionais

Para identificar que técnicas de programação devem ser adotados como boas práticas para a Computação Verde, foram implementados os cenários propostos na seção 4, tendo seus gastos energéticos quantizados através da ferramenta Joulemeter apresentada neste trabalho.

Os cenários propostos foram implementados utilizando a linguagem Java. Todos os experimentos computacionais foram realizados em um computador com sistema operacional Linux Ubuntu 9.04 e com processador Intel Core 2 Quad 2.33 Ghz com 2 GB de memória RAM.

Cenário 1: Uso de dispositivos de Entrada e Saída

Para o primeiro cenário foram implementadas duas estratégias para trabalhar com a escrita da memória secundária. A primeira estratégia escreve diretamente no disco sempre que pedido e a segunda usa um buffer de tamanho 8KB para armazenar os pedidos e apenas quando cheio, descarregá-lo no dispositivo. Os arquivos gerados nos teste possuem tamanho 31.250 KBytes e cada escrita individual possui 32 Bytes.

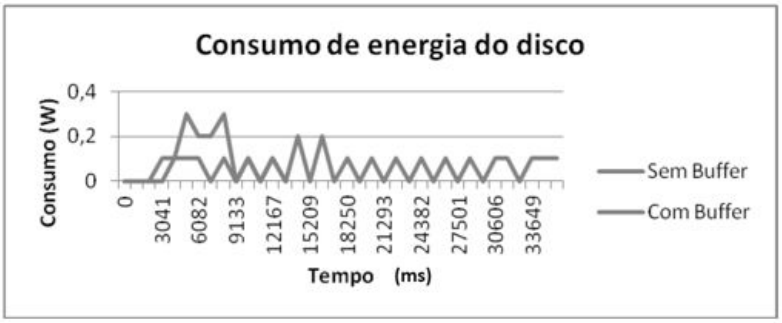

Figura 3. Consumo comparativo de disco entre programas com e sem uso de buffer.

Na Figura 3 é apresentado um gráfico de consumo para as duas estratégias implementadas. Como pode ser observado, a solução com buffer concluiu a tarefa mais rapidamente (após 12 segundos) do que a forma sem buffer (após 33 segundos). Entretanto é importante observar que com buffer o disco atinge um consumo máximo maior, chegando a 0,3 Watts, enquanto que sem buffer o máximo atingido foi 0,2 Watts. O consumo acumulado da solução com buffer foi de 1,5W, enquanto que sem buffer o mesmo foi de 2,1W, mostrando que vale a pena conhecer como os arquivos são salvos e a importância do uso de buffer, pois ao adotá-lo a aplicação não espera pela escrita no disco para continuar processando. Isso faz com que o desempenho seja bem mais elevado.

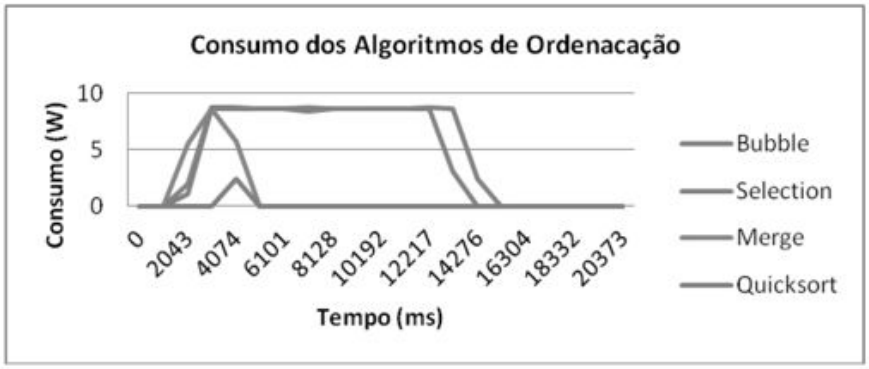

Figura 4. Consumo comparativo de CPU entre diferentes algoritmos de ordenação

Cenário 2: Eficiência de algoritmos

No segundo experimento foram comparados os consumos energéticos de quatro algoritmos clássicos de ordenação. O mesmo conjunto de dados foi passado para todos os algoritmos, para isso, foi gerado um vetor aleatório com distribuição normal contendo 1 milhão de elementos ($n = 10^6$) a ser ordenado.

A Figura 4 apresenta os resultados deste cenário, onde é possível observar que, com exceção do Quicksort, todos os algoritmos atingiram um máximo de consumo de 8,4W, porém o Bubblesort e o Selectionsort permaneceram mais tempo consumindo nesse limite, gerando algoritmos menos energeticamente eficientes.

O Mergesort atingiu bom resultado, finalizando rapidamente e consumindo energia por menos tempo. Já o Quicksort, como é conhecido por seu alto desempenho médio, também se mostrou bastante eficiente no que trata de consumo de energia. Apesar destes últimos algoritmos terem a mesma eficiência média, a diferença de consumo energética de seus resultados pode ser explicada pela característica do vetor gerado, que pode ter favorecido o Quicksort, já que o mesmo, diferente do Mergesort, é uma implementação instável, ou seja, dependente da distribuição dos elementos na entrada.

Os consumos acumulados dos algoritmos de ordenação foram: Bubblesort: 91,1W; Selectionsort: 100W; Mergesort: 19,9W e Quicksort: 2,4W. Esses resultados mostraram o quão importante é utilizar algoritmos mais eficientes, não apenas por executarem mais rapidamente, mas também por saber que os mesmos consomem menos energia.

Cenário 3: Algoritmo Iterativo X Algoritmo Recursivo

No último teste foram comparados dois algoritmos de percurso sobre árvore binária em pré-ordem, porém um com abordagem iterativa e outro recursivo. O mesmo conjunto de dados foi passado para ambos, para isso, foi gerado um vetor ordenado, pré-requisito do funcionamento deste tipo de algoritmo, contendo 20 milhões de elementos ($n = 2*10^7$) a serem percorridos.

O resultado deste cenário é apresentado na Figura 5. Os consumos de energia acumulados na execução dos algoritmos foram: iterativo, 49.3W; recursivo: 66W.

O algoritmo recursivo, apesar de mais legível e intuitivo para ser programado em aplicações que trabalham com busca em vetor, árvore e grafo, se mostrou maior consumidor de energia. A razão desse aumento no consumo de energia pode se dar pelo processamento extra e maior uso de memória pelo Sistema Operacional para manter a pilha de chamadas recursivas armazenadas.

6. Considerações Finais e Trabalhos Futuros

Neste trabalho, foi abordado um estudo da Computação Verde, apresentando as principais áreas de atuação para a construção de sistema sustentáveis. Percebe-se que as principais contribuições se dão na evolução da tecnologia do hardware e da criação de softwares de gerenciamento no uso destes dispositivos físicos. Foi proposto neste trabalho a identificação de técnicas de programação que tenha como características a eficiência energética em sua execução. Para isso, foram analisadas várias ferramentas de monitoramento energético e adotada a ferramenta Joulemeter em nossos experimentos.

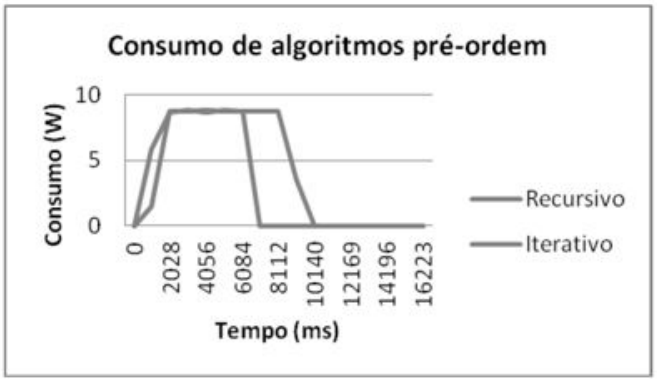

Figura 5. Comparativo entre um algoritmo recursivo e um iterativo.

Com o objetivo de identificar práticas de programação com menor consumo energético, foram propostos cenários de teste, cada um comparando técnicas que tenham a mesma função. No primeiro cenário, foram testadas duas estratégias de escrita no dispositivo de E/S. Os resultados mostraram que o uso de um *buffer* intermediário, em memória primária, que armazene uma quantidade de chamadas para realizá-las de uma vez no dispositivo, reduz o consumo de energia em mais de 25%.

No segundo cenário onde foi realizada uma relação entre a complexidade computacional de algoritmos e o consumo energético, os resultados mostraram que

algoritmos com eficiência assintótica firme (notação Θ) com menores valores têm maior eficiência energética que algoritmos com maior valor assintótico. Como exemplo, temos o algoritmo Mergesort, cuja eficiência é $\Theta (n \log n)$, que tem um ganho de 78,15% no consumo de energia comparado ao Bubblesort, cuja eficiência é $\Theta (n^2)$.

No último cenário, foi proposta a comparação de um algoritmo iterativo com outro recursivo. Para que a eficiência assintótica dos algoritmos não tenha influência na comparação, foi adotado um algoritmo que mesmo com a implementação de técnicas distintas de programação tivessem o mesmo valor assintótico, neste caso, eficiência linear. Os resultados mostram que o algoritmo iterativo apresentou um ganho energético de 25,3% com relação ao consumo de energia do algoritmo recursivo.

Portanto, para trabalhar na construção de sistemas computacionais mais sustentáveis, é possível adotar além das tradicionais evoluções no uso dos hardwares, técnicas de programação verde, como as técnicas analisadas neste trabalho.

Como trabalhos futuros, é planejado estender a identificação de técnicas de programação verde, criando uma base de decisões que servirão para identificar novas práticas a serem adotadas nos cursos de Engenharia de Software, permitindo ao usuário criar soluções eficazes e ao mesmo tempo com eficiência energética.

Referências Bibliográficas

Albiero, F.W. (2010) "Monitoramento e Avaliação do Consumo Energético em Função do Poder Computacional". Trabalho de Graduação. Disponível em:< http://www-app.inf.ufsm.br/bdtg/arquivo.php?id=135&download=1 >, acessado em 21 de março de 2012.

Ascencio, A. F.; Araújo, G. S. (2011) "Estruturas de Dados: algoritmos, análise da complexidade e implementações em Java e C/C++". São Paulo: Pearson.

Barbosa, F.P.; Charão, A. S. (2009) "Grid Computing e Cloud Computing – Uma Relação de Diferenças, Semelhanças, Aplicabilidade e Possibilidades de Cooperação entre os dois Paradigmas". Disponível em: < www-usr.inf.ufsm.br/~andrea/elc888/artigos/artigo4.doc >, acessado em 14 de abril de 2012.

Dietrich, J.; Schmidt, R. (2007) "O Datacenter Verde". IBM. Disponível em: < http://www.ibm.com/br/services/gts/pdf/Datacenter_verde.pdf>, Acessado em 28 de abril de 2012.

Everest. 2007. Disponível em: < http://www.hardware.com.br/comunidade/guia-basico/746737/ >, acessado em 25 de março de 2012.

Fernández, A.; Marcelino, J.; Marques, P. Cloud Computing. 2011. Disponível em: < http://www.marcasepatentes.pt/files/collections/pt_PT/1/300/301/Cloud%20Computing.pdf >, acessado em 10/04/2012.

Harmon, R.; Demirkan, H.; Auseklis, N.; Reinoso, M. (2010) "From Green Computing to Sustainable IT: Developing a Sustainable Service Orientation". Proceedings of the 43rd Hawaii International Conference on System Sciences. IEEE.

HWMonitor. 2012. Disponível em: < http://hwmonitor.software.informer.com/ >, acessado em 25 de março de 2012.

Joulemeter. 2011. Disponível em: <http://research.microsoft.com/en-us/projects/joulemeter/default.aspx>, acessado em 25 de março de 2012.

Li, Q.; Zhou, M. (2011) "The Survey and Future Evolution of Green Computing". International Conference on Green Computing and Communications. IEEE.

Mór, S.D.K.; Alves, M.A.Z.; Lima, J.V.F.; Maillard, N.B.; Navaux, P.O.A. (2010) "Eficiência Energética em Computação de Alto Desempenho: Uma abordagem em Arquitetura e Programação para Green Computing". Disponível em: < http://www.inf.pucminas.br/sbc2010/anais/pdf/semish/st01_01.pdf >, acessado em 10 de abril de 2012.

Prush, K. (2011) "Green Computing Algorithmics". 52nd Annual Symposium on Foundations of Computer Science. IEEE.

Rolt, J.; Bonin, L.C.; Cittadin, L.; Mattei, L. F. (2010) "TI Verde: Uma nova forma de evoluir com preocupação ambiental e sustentável". Relatório Científico. Disponível em: <http://www.ecolmeia.org.br/blog/wp-content/uploads/2010/08/TI-Verde-Inst-Gaidzinski.pdf>, acessado em 02 de abril de 2012.

Salgado, G. T. (2011) "Estudo sobre o Impacto Energético de Máquinas Virtuais em um Sistema Computacional Físico". Trabalho de Graduação. Disponível em:< http://www.lume.ufrgs.br/handle/10183/31051>, acessado em 21 de março de 2012.

SBC. (2006) "Grandes Desafios da Pesquisa em Computação no Brasil". Disponível em:<http://www.sbc.org.br/index.php?option=com_jdownloads&Itemid=195&task=finish&cid=11&catid=50>, acessado em 20 de março de 2012.

Silva, M.R.P.; Zaneti, B.G.; Zago, M.G.; Souza, A.N. (2010) "TI Verde – Princípios e Práticas Sustentáveis para Aplicação em Universidades. Disponível em: <http://www.labplan.ufsc.br/congressos/III%20SBSE%20-202010/PDF/SBSE2010-0085.PDF>, acessado em 29 de março de 2012.

Vieira, A.; Debastiani, D.; Agostini, L.; Marques, F.; Mattos, J. (2012) " Performance and Energy Consumption Analysis of Embedded Applications based on Android Platform". In: Simpósio Brasileiro de Engenharia de Sistemas Embarcados – SBESC 2012. Natal – RN.

Vykoukal, J.; Wolf, M.; Beck, R. (2009) "Does Green IT Matter? Analysis of the Relationship between Green IT and Grid Technology from a Resource-Based View Perspective". Disponível em: <http://ade.se/skola/ht10/infn14/articles>, acessado em 23 de março de 2012.