Eficiência Energética em Computação de Alto Desempenho: Uma Abordagem em Arquitetura e Programação para Green Computing

Stéfano D. K. Mór, Marco A. Z. Alves, João V. F. Lima, Nicolas B. Maillard, Philippe O. A. Navaux

[1]Instituto de Informática – Universidade Federal do Rio Grande do Sul (UFRGS)
Caixa Postal 15.064 – 91.501-970 – Porto Alegre – RS – Brazil

{sdkmor, marco.zanata, jvlima, nicolas, navaux}@inf.ufrgs.br

Abstract. *This paper presents an overview about the demand on high-performance, energy-efficient and low-power parallel platforms over an architecture and parallel programming oriented view-point. We present the position of the Parallel and Distributed Processing Group (GPPD) of Federal University of Rio Grande do Sul (UFRGS) about the relationship between a major factor in making "green computing", the low-power, with high computational efficiency. This context was related with the challenges of Brazilian research on Computing proposed by the Brazilian Computing Society (SBC) in 2006. We defend that parallelism is a key factor at the distributed processing of large volumes of data and that this process can be accomplished with a scalable and efficient consumption of energy. We present a context to some previous work where one can manage power usage through the maximum use of active processors (at programming level) and, also, through reducing power consumption of components in a parallel environment (at architecture level). In the end, we present our prospects for the use of parallelism as a tool in obtaining a rational use of energy consumption over next years.*

Resumo. *Este artigo apresenta uma visão sob o ponto de vista de arquitetura de computadores e programação paralela frente à demanda por plataformas paralelas de alto desempenho e baixo consumo energético. Apresenta-se a visão do Grupo de Processamento Paralelo e Distribuído (GPPD) da UFRGS sobre a relação entre um dos principais fatores na construção de "sistemas verdes" (green computing), o baixo consumo de energia elétrica, com a alta eficiência computacional. Esse contexto é relacionado com os desafios da pesquisa brasileira na Computação propostos pela Sociedade Brasileira de Computação em 2006. Mostramos 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. Apresentamos trabalhos que conseguem regular o consumo de energia dispendido através do máximo aproveitamento de processadores ativos (no nível de programação) e, também, estudos que visam a redução do consumo de potência dos componentes presentes em um ambiente paralelo (no nível de arquitetura). Ao final, apresentamos nossas perspectivas para o uso de paralelismo como ferramenta na obtenção de um consumo racional de energia nos próximos anos.*

1. Introdução

Em 2006, a Sociedade Brasileira de Computação (SBC) elegeu cinco grandes desafios da pesquisa em computação no Brasil para os próximos dez anos:

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

Passados quatro anos, a comunidade científica brasileira caminha a passos largos, rumo a uma realidade compatível com esses objetivos. No entanto, existem avanços necessários que ainda precisam ser alcançados nas mais diversas áreas da Computação.

Este artigo apresenta a posição do Grupo de Processamento Paralelo e Distribuído (GPPD) da UFRGS frente à demanda por plataformas de alto desempenho paralelas e ecologicamente sustentáveis. O principal ponto explorado neste trabalho é mostrar a relação entre um dos principais fatores na construção de "sistemas verdes", o baixo consumo de energia elétrica, com a alta eficiência computacional. Além disso, o artigo apresenta, na visão do grupo, como essa demanda está relacionada com os desafios em Computação apresentados.

Um dos focos do trabalho realizado pelo GPPD é a integração eficiente de técnicas avançadas de programação paralela com uma variedade de importantes otimizações arquiteturais em máquinas multiprocessadas. Desse modo, o grupo espera prover um ambiente homogêneo para a execução de programas paralelos, que compute um resultado no menor tempo possível, utilizando pouca memória e fazendo uso intensivo e otimizado de recursos ociosos. Acreditamos que resultados expressivos nesse tópico contribuam fortemente na resolução dos desafios 1 e 5, pois (a) o processamento de alto desempenho é vital para o tratamento distribuído de grandes volumes de dados em tempo reduzido e; (b) o projeto inteligente de algoritmos e arquiteturas paralelas, dada a atual tendência de adoção generalizada de paralelismo tanto em *software* quanto em *hardware*, é central na construção de tecnologias escaláveis.

Contextualizando o atual cenário da área, a corrida pelo aumento da frequência de *clock* e aprofundamento de técnicas de paralelismo no nível de instrução (ILP – *Instruction Level Paralelism*) [Smith and Sohi 1995] nos microprocessadores comerciais (disputada pelos principais fabricantes nas décadas de 1980 e 1990) deu lugar à corrida pelo aumento do paralelismo no nível de *threads*, com a introdução de uma quantidade elevada de núcleos de processamento em um mesmo circuito integrado [Olukotun et al. 1996], nomeados *chips multi-core*. Neste sentido, existem diversas pesquisas sobre formas de comunicação entre os diversos núcleos, seja através de uma rede intra-*chip* [Bjerregaard and Mahadevan 2006] [Freitas et al. 2007] [De Micheli and Benini 2006], ou por memória compartilhada, onde existem pesquisas sobre formas de melhor organizar as memórias *cache* desses núcleos de processamento [Alves et al. 2009] [Marino 2006], além de possíveis alterações arquiteturais, como no

caso das memórias cache de arquitetura não-uniforme (NUCA) [Bardine et al. 2008] [Kim et al. 2002].

O uso de agregados (*clusters*) de computadores [Navaux and DeRose 2003], amplamente utilizados na computação de alto desempenho, cada vez mais se beneficia da introdução de novas e mais rápidas interconexões, ao mesmo tempo em que a utilização de *chips multi-core* como nós nesses agregados cria um ambiente paralelo multi-nível. Com tantos núcleos de processamento operando de maneira concorrente e considerando o movimento mundial rumo a um desenvolvimento tecnológico sustentável, torna-se imperativo desenvolver sistemas paralelos que, além de prover alto desempenho, endereçando os desafios propostos pela SBC, consigam fazê-lo com um baixo consumo de energia elétrica, contribuindo, assim, para o melhor aproveitamento dos recursos naturais do planeta.

O restante do artigo esta organizado da seguinte maneira: A Seção 2 mostra a visão do GPPD de como a introdução de paralelismo nos diferentes níveis de computação contribui significativamente para *green computing* e enumera trabalhos relacionados na mesma linha de pensamento. A Seção 3 discute o papel de alguns avanços obtidos pelo grupo no uso de *software* paralelo eficiente para o processamento *on-line* e distribuído de grandes quantidades de dados e como isso impacta indiretamente na questão ambiental. A Seção 4 apresenta nossa visão em relação à importância da arquitetura de processadores paralelos para a computação escalável e de baixo consumo de energia elétrica, ilustrando ganhos obtidos nesses setores com resultados previamente publicados. Por fim, a Seção 5 apresenta conclusões e observações importantes sobre o tema e as direções de trabalhos futuros do grupo.

2. Paralelismo e Computação Verde

Esta seção é destinada a mostrar a ligação muito próxima entre a área de Computação Verde e a utilização de máquinas paralelas como supercomputadores provedores de serviços. *E.g.*, [Wang 2007] sugere que a ligação entre *Green Computing* e o baixo consumo energético é imediata e que, aos poucos, a indústria de Tecnologia da Informação (TI) se foca no desenvolvimento de soluções com menor consumo energético como prioridade. Além disso, [Feng et al. 2008] argumenta que a área, antes subestimada, se mostra central no desenvolvimento de sistemas computacionais de grande capacidade. Estes sistemas, que visam atender a uma larga demanda de dados, incorporaram uma série de atributos desejáveis com o passar dos anos, como, *e.g.*, pervasividade, confiabilidade, transparência e manutenabilidade, *etc*. Dado o poder computacional necessário para este tipo de processamento sobre dados massivos, o uso de supercomputadores e *data centers* é a via que surge naturalmente na implementação destes sistemas. Em ambos os casos, a introdução de paralelismo é o fator crítico na obtenção de processamento altamente eficiente e escalável. Esta visão do papel das máquinas paralelas como foco do desenvolvimento em Computação Verde é compartilhada por [Vykoukal et al. 2009], onde se apresenta uma discussão mais detalhada sobre o tema.

O GPPD atua de duas maneiras centrais para reduzir o consumo energético com a introdução de paralelismo: (a) introduzir técnicas arquiteturais mais eficientes, do ponto de vista de consumo de energia, pelo aumento do desempenho e também no contexto de projetos arquiteturais de menor consumo de potência; (b) aumentar ao máximo o uso de processadores que já estão ativos, diminuindo sua ociosidade, evitando que processadores

extras sejam usados e aproveitando a energia que estes já estão consumindo.

O aumento da eficiência energética é importante em todo contexto computacional, independente do *hardware* adotado. Ou seja, para todas arquiteturas e implementações físicas de máquinas paralelas, as quais podem apresentar diferentes consumos de potência, a eficiência energética é um fator crucial a ser analisado. Nesse sentido, a programação paralela tem papel fundamental na melhoria da eficiência, utilizando melhor os recursos computacionais e aumentando o desempenho das aplicações. Por outro lado, a organização e arquitetura do processador paralelo podem ajudar ainda mais na redução do consumo energético, com dispositivos mais eficientes e novas técnicas de aumento de desempenho. Além disso, existe a atuação sobre o consumo de potência, onde a arquitetura, seja através de componentes mais sofisticados, seja pelo desligamento de unidades que não estão sendo utilizadas em uma certa porção de tempo, poderá colaborar com a redução do consumo de potência .

Como exemplo que reforça nosso ponto, consideremos a Fig. 1. A figura apresenta uma execução hipotética de uma aplicação paralela cujo tempo de execução para o caso de uma *thread* é igual a uma hora. Cada curva indica o consumo de energia, dado em Joules (J), considerando que a arquitetura do processador fosse capaz de desligar de 0% a 100% os núcleos não utilizados. Os valores se baseiam em uma máquina com dois processadores Xeon 5530 Quad-Core de arquitetura Nehalem, cujo consumo de potência é igual a 80 W em cada processador. O sistema completo, em estado *idle*, consome cerca de 160 W. Através da Fig. 1 podemos verificar a importância da introdução de melhorias na programação paralela; mesmo sem mudança arquiteturais, há um claro aumento de eficiência energética. A figura ilustra, também, as possíveis melhoras na eficiência e também no consumo de energia caso possamos diminuir uma porcentagem da potência empregada em núcleos que não estão sendo utilizados em um dado momento. Essa redução de consumo nos núcleos ociosos é importante, uma vez que as aplicações reais nem sempre são facilmente paralelizadas ou apresentam uma versão paralela nativa.

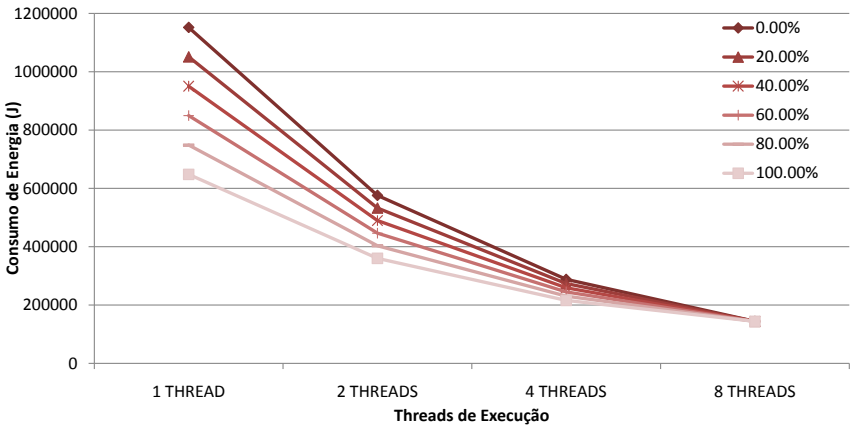


Figura 1. Consumo energético (eixo vertical) para uma aplicação com diversos níveis de paralelismo (eixo horizontal) e diversos níveis de desligamento de recursos ociosos (diferentes curvas).

O trabalho atual do grupo compara-se com iniciativas de outros grupos de pesquisa que abordam o tema de maneira semelhante. Há uma proliferação de trabalhos que, tal qual nossos esforços, enfocam o emprego de máquinas de alto-desempenho, intrinseca-

mente paralelas, como provedoras de serviços e que atuam de maneira transparente, com foco em eficiência e controle do consumo energético. *E.g.*, [Binder and Suri 2009] propõe um algoritmo de alocação e despacho de tarefas que minimize o uso de servidores ativos necessários, gerando economia de recursos nos mesmos moldes dos que são apontados pela Fig. 1. Na mesma linha, [Torres et al. 2008] apresenta uma combinação entre compressão de dados e atendimento seletivo de requisições em um *data center* de modo que 20% menos servidores são necessários para realizar uma determinada tarefa. Em ambos os casos, as técnicas referidas abrangem *hardware* e *software* e se centram na desativação de recursos ociosos em prol do baixo consumo energético. Essa abordagem multi-nível *hardware*/*software* pode ser vista em [Feng 2003], que apresenta a combinação de uma máquina construída para ser eficiente e energeticamente econômica e um *framework* para consulta paralela de dados via programação com troca de mensagens, também projetada para minimizar os recursos ociosos. É esse viés multi-camada da computação paralela que apresentamos nas próximas seções.

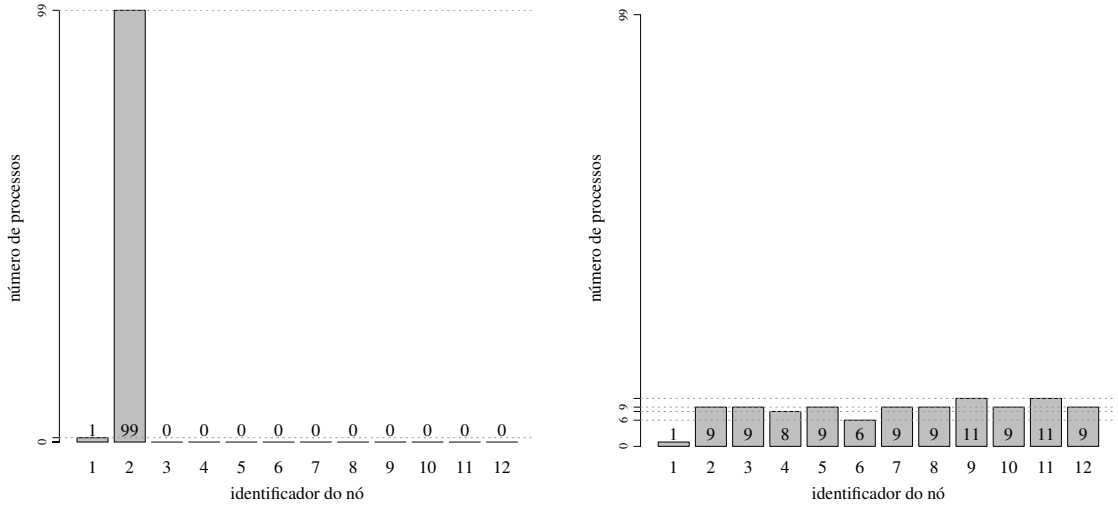
3. O Papel da Programação Paralela

Conforme observado anteriormente, uma diminuição no tempo de execução impacta diretamente na quantidade de energia consumida em uma computação paralela. [Asanovic et al. 2009] apresenta várias considerações importantes sobre o papel da Programação Paralela no enfrentamento dos problemas que aparecerão nos próximos anos. O objetivo do uso de Programação Paralela para economia de recursos é, ao nosso ver, obter um aumento de *speedup* $\left(\frac{\text{tempo total}}{\text{\# processadores}}\right)$ [Dongarra et al. 2003] através de técnicas de alocação de tarefas em recursos (*i.e.*, técnicas de escalonamento de tarefas paralelas) e do controle da granularidade (*i.e.*, quantidade de trabalho sequencial em uma tarefa paralela) [Foster 1995]. As observações dessa Seção consideram o cenário onde se leva em conta um subconjunto de processadores que está ativo e consumindo a mesma quantidade de energia continuamente.

A idéia geral sobre o papel da programação paralela no aproveitamento dos recursos energéticos é que um algoritmo paralelo bem programado é capaz de utilizar os recursos à disposição de maneira eficiente, fazendo com que um número mínimo de unidades de processamento que estão ligadas fiquem ociosas, aumentando, assim, o *speedup*. Vários trabalhos no grupo abordam o ganho de *speedup* através do escalonamento eficiente de tarefas sobre modelos de programação paralela, sobretudo [Mór and Maillard 2009, Cera et al. 2007, Pezzi et al. 2007].

Imaginemos uma máquina do tipo *Multiple Instruction, Multiple Data* (MIMD) [Flynn 1972], um *cluster* de computadores onde cada nó é uma unidade computacional completa, equipada com processadores (possivelmente 1, possivelmente muitos) e memória privativa . Durante o tempo alocado, todos os nós ficam ativos. Vamos considerar que o ganho de energia almejado deve ser obtido exclusivamente pelo ganho de *speedup* (todos os processadores considerados estão operando com a máxima frequência de *clock* disponível). Consideremos também, que uma tarefa paralela leva, no máximo, α ciclos de *clock* de um núcleo de processamento para ser consumida e que durante um ciclo de *clock* o processador consome β *Joules* de energia. Vejamos, então, a Fig. 2, que mostra o resultado, em termos de alocação, de duas estratégias para escalonamento de tarefas criadas em tempo de execução: *round-robin* e *roubo de tarefas*.

A máquina onde foram realizados os experimentos é um *cluster* de 14 nós DELL PowerEdge 1950, onde cada nó tem dois processadores Intel Xeon E5310 Quad Core de 1.60 GHz. Cada processador tem 2×4MB de memória *cache* (1066 *Front Side Bus*). A interconexão dos nós é feita por um *switch* 3Com modelo 2816 via Gigabit Ethernet.



(a) Escalonamento *round-robin*. (b) Escalonamento com *roubo de tarefas*.

Figura 2. Balanceamento de carga de tarefas por nó usando dois algoritmos de escalonamento: *round-robin*, (a), e *roubo de tarefas*, (b).

Na Fig. 2, uma tarefa raiz cria outras 99 tarefas, dinamicamente. Cada tarefa dinâmica, um processo do SO, é criada com um intervalo de $1s$ em relação à anterior e imprime em `stdout` a URL do nó que a está executando. O número sobre cada coluna é a replicação do valor correspondente no eixo vertical.

Através da Fig. 2 é possível aproximar os custos das diferentes abordagens de escalonamento em termos de tempo e consumo de energia. A cada α ciclos de *clock* uma tarefa é consumida, de modo que todas as tarefas que estejam no mesmo nível (valor do eixo y) são executadas em paralelo; esse é o tempo de execução de uma tarefa. A quantidade de energia dissipada é obtida multiplicando o tempo de execução pela quantidade de energia gasta em cada ciclo (β) pelo número de processadores participantes (1 ou 12), conforme a Tabela 1.

Tabela 1. Análise parcial do impacto do balanceamento de carga em sistemas paralelos em relação ao tempo e consumo de energia.

estratégia	tempo	consumo de energia
1 processador	$100\text{s} \times \alpha$	$100\text{J} \times \alpha\beta$
12 processadores com *round robin*, Fig. 2(a)	$99\text{s} \times \alpha$	$1.188\text{J} \times \alpha\beta$
12 processadores com *roubo de tarefas*, Fig. 2(b)	$11\text{s} \times \alpha$	$132\text{J} \times \alpha\beta$

O ponto a ser mostrado é que a introdução de um grau ótimo de paralelismo no nível de programação pode proporcionar uma grande economia de energia. Cabe aos programadores definir um nível satisfatório de paralelismo de acordo com sua aplicação.

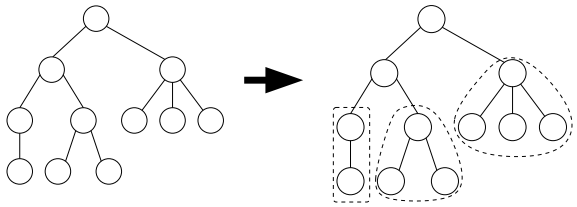
E.g., ao escalonar tarefas em 12 processadores com *roubo de tarefas* há uma redução de aproximadamente 89% no tempo de execução em relação a um processador, enquanto o consumo de energia aumenta aproximadamente 32%. Se, ao invés de 12 processadores, 6 processadores fossem usados, o tempo diminuiria em 78% enquanto o consumo de energia *diminuiria* em 34%. Esses valores podem ser variados de acordo com as necessidades da aplicação sem necessitar de um investimento significativo na parte de *hardware*. Algoritmos de escalonamento produzem redução no tempo de execução e no consumo de energia elétrica, contanto que um número de processadores adequado seja utilizado.

Dentro da Programação Paralela, entendemos que o *grão* é, basicamente, uma quantidade de trabalho contíguo que executará em algum processador; um grão não pode ser "quebrado" em partes menores executadas em paralelo, tendo de ser executado sequencialmente. O controle de granularidade é, então, determinar um *tamanho de grão* que maximize o paralelismo e diminua o *overhead* devido à comunicação e sincronização entre processadores; um grão grande oferece baixo paralelismo e baixo *overhead* de comunicação, enquanto um grão pequeno configura-se inversamente. Acreditamos que o controle eficiente da *granularidade* de aplicações paralelas seja uma ferramenta útil para o processamento escalável de grandes volumes de dados e, dada a sua característica de adaptabilidade, eficiente em minimizar o consumo de energia elétrica. O grupo possui trabalhos que tratam do uso do controle de granularidade para a melhor alocação dos recursos disponíveis como, *e.g.*, [Lima and Maillard 2009].

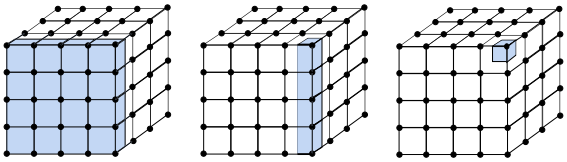
A Fig. 3 mostra o emprego do controle de granularidade sob duas abordagens: granularidade lógica, que configura o tamanho do conjunto de operações sobre um dado, e a granularidade de dados, que configura o tamanho do dado que será alvo do conjunto de operações sequenciais ou particionado em sub-estruturas. Ambas abordagens podem ser aplicadas separadamente ou em conjunto. A Fig. 3(a) mostra a variação de granularidade em um programa recursivo que executa em forma de árvore; na árvore da esquerda as folhas são distribuídas para a computação em paralelo, enquanto na árvore da direita acontece o mesmo com um agrupamento destas. Já a Fig. 3(b) mostra o controle de granularidade em uma decomposição de domínio em três dimensões; a distribuição dos dados em processadores é feita alocando-se em planos, retas ou pontos do cubo representado, variando-se a quantidade de operações exercida por cada processador.

Queremos apontar, em especial, a utilidade do controle do tamanho de grão usado para definir, dinamicamente, a representação interna do fluxo de processamento de uma tarefa, *i.e.*, a utilidade da decisão se representar uma tarefa como um *processo* de SO (que tem maior *overhead* de comunicação) ou como uma *thread* (que tem menor *overhead* de comunicação). A grosso modo, quando uma tarefa é criada como um processo, ela pode ser disparada remotamente; todas as dependências em nível de máquina são privativas e podem ser transportadas para outro *host*. Quando uma tarefa é criada como uma *thread* ela deve necessariamente ficar na mesma máquina da tarefa criadora; a área de código compartilhada impede a distribuição remota da *thread* recém-criada.

Ao criar um número razoável de tarefas como *threads* abre-se mão do paralelismo, mas ganha-se em desempenho, tanto na criação da nova tarefa quanto na redução da ordem de magnitude do *overhead* de comunicação. Ao se optar por processos, ganha-se paralelismo real, ao custo do aumento de *overhead* de comunicação [Balaji et al. 2008]. Disparar um tarefa recém criada como um processo ou *thread* é uma decisão que pode ser



(a) Exemplo de controle de granularidade lógica.



(b) Exemplo de controle de granularidade de dados.

Figura 3. Controle de granularidade lógica e de dados.

feita de maneira estática, em tempo de compilação, mas é uma solução pouco escalável e generalizável, adequada apenas em casos específicos. Enxergamos a tomada de decisão em tempo de execução como uma abordagem mais compatível com os desafios em pesquisa propostos pela SBC.

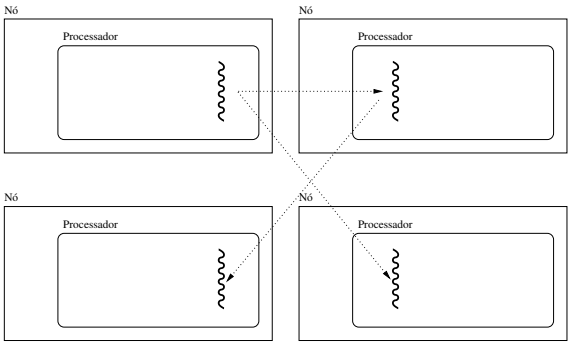
Um *middleware* para o controle dinâmico da granularidade deve levar em conta vários fatores, como carga de tarefas na máquina local, largura de banda disponível para a comunicação e suporte a paralelismo em nível de *threads* por parte do processador, *etc.*. O ponto central é que o controle de granularidade, além de proporcionar adaptabilidade, compatível com o desafio #5, pode ser utilizada como ferramenta no controle do gasto energético na computação de um dado problema, sem prejuízo do ganho de velocidade esperado.

Para reforçar nosso ponto, consideremos a Fig. 4. Nela, cada linha em curva é um fluxo de instruções; a linha mais grossa é uma thread que cria outras threads ou outros processos e a linha mais fina é uma *thread* que cria somente outras *threads*. A seta pontilhada representa a criação remota de tarefas, enquanto a seta tracejada representa a criação local de tarefas. Na Fig. 4(a) não há controle de granularidade; a cada criação de uma nova tarefa um novo processo é alocado em uma máquina remota, que é o comportamento padrão do *middleware* mais popular para computação paralela em *clusters* de computadores, o MPI. Já a Fig. 4(b) mostra o comportamento de uma versão do MPI modificada para realizar o controle de granularidade. Vamos considerar que as medições ocorreram em ω ciclos de *clock* (ω constante); o comparativo entre ambos pode ser visto na Tabela 2.

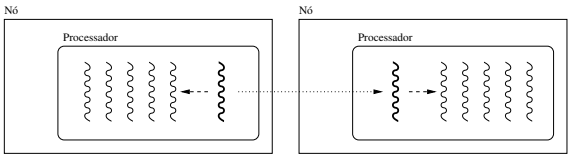
Tabela 2. Comparativo de criação de tarefas em ω ciclos de *clock* com e sem controle de granularidade.

estratégia	# tarefas criadas	# proc. utilizados	consumo
sem controle de granularidade	4	4	$4\omega\beta$J
com controle de granularidade	12	2	$2\omega\beta$J

O número de tarefas criadas no mesmo instante de tempo foi medido no expe-



(a) Tarefas como processos (requer migração de processador).



(b) Tarefas como *threads* ou processos (migração de processador ocorre ocasionalmente).

Figura 4. Aproveitamento de processadores disponíveis em ω ciclos de *clock* sem o uso de controle de granularidade (a) e com o controle (b).

.

rimento que produziu a Fig. 5, a execução do algoritmo de ordenação por intercalação (*Mergesort*), comparados uma execução com controle de granularidade e uma execução sem o mesmo. Além do uso de um menor número de processadores, o controle dinâmico de granularidade consegue um aumento drástico no desempenho de algoritmos paralelos, fator que, conforme já discutido, contribui para a redução do consumo de energia.

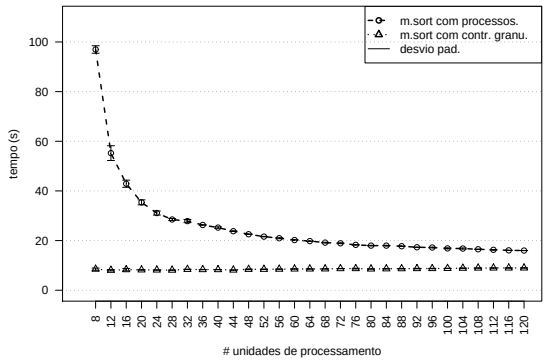


Figura 5. Tempos com controle de granularidade para o *Mergesort*.

4. O Papel das Arquiteturas Paralelas

No atual contexto de inovações em *multi-core*, em que as novas tecnologias de integração estão fornecendo um número crescente de transistores por *chip*, o estudo de técnicas de aumento de vazão de dados é de suma importância para os atuais e futuros processadores *multi-core* e *many-core*.

Com a contínua demanda por desempenho computacional, as memórias *cache* vêm sendo largamente adotadas nos diversos tipos de projetos arquiteturais de computadores. Os atuais processadores disponíveis no mercado apontam na direção do uso de memórias *cache* L2 compartilhadas. No entanto, ainda não está claro quais os ganhos e custos inerentes desses modelos de compartilhamento da memória *cache*. Assim, nota-se a importância de estudos que abordem os diversos aspectos do compartilhamento de memória *cache* em processadores com múltiplos núcleos.

O trabalho que foi desenvolvido no grupo [Alves 2009], teve como objetivo avaliar diferentes compartilhamentos de memória *cache*, modelando e aplicando cargas de trabalho sobre as diferentes organizações, a fim de obter resultados significativos sobre o desempenho e a influência do compartilhamento da memória *cache* em processadores *multi-core*. Após definir e ponderar sobre os fatores para decisão de escolha por determinada carga de trabalho, foi escolhido o conjunto de aplicações paralelas NAS-NPB para ser utilizada como carga de trabalho real entre os diversos experimentos. Uma vez que esse *benchmark* apresenta diversas aplicações relacionadas a métodos numéricos de simulações aerodinâmicas para computação científica que foram projetados para comparar o desempenho de computadores paralelos.

Esses aplicativos são formados por *kernels* e problemas de simulação de dinâmica de fluídos computacionais (CFD - *Computational Fluid Dynamics*) derivados de importantes aplicações das classes aerofísicas, de maneira que as simulações de CFD reproduzem grande parte dos movimentos de dados e computação encontrada em códigos CFD completos [Jin et al. 1999].

O *benchmark* utilizado neste estudo foi a versão NPB-OpenMP 3.3 que possui as seguintes aplicações: BT (*Block Tridiagonal*), CG (*Conjugate Gradient*), MG (*Multigrid*), EP (*Embarassingly Parallel*), SP (*Scalar Pentadiagonal*), LU (*Lower and Upper triangular system*), IS (*Integer Sort*), FT (*Fast Fourier Transform*), UA (*Unstructured Adaptive*) e DC (*Data Cube*). Todas as aplicações executam operações de ponto-flutuante de precisão dupla (64 bits), foram implementados em linguagem Fortran ou C e paralelizado utilizando OpenMP.

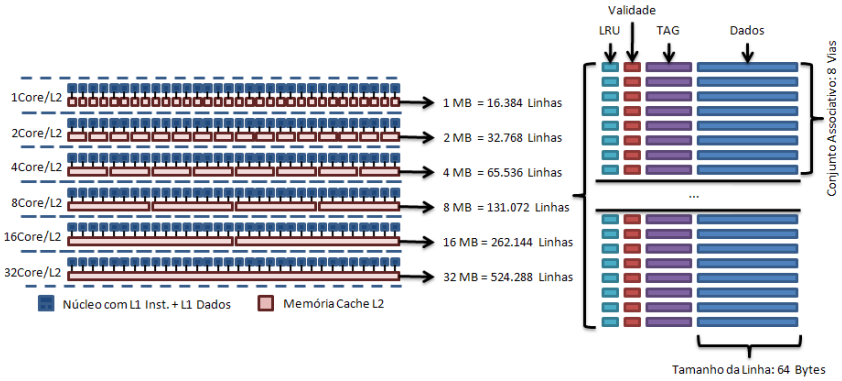


Figura 6. Experimento base para o estudo sobre o compartilhamento de memórias *cache* L2.

O primeiro experimento, que serviu como base para os demais, é o que avaliou o compartilhamento da memória *cache* L2. Nestes testes foram fixados o tamanho total de

memória *cache* em 32 MB (somando todos os bancos de memória *cache* L2), e a partir da configuração de um núcleo de processamento por memória *cache* L2, foi variando o número de núcleos compartilhando o mesmo banco de memória *cache*, como é ilustrado na Figura 6, onde é possível ver a nomenclatura para cada um dos testes com diferentes números de núcleos por memória *cache* L2, além dos demais parâmetros fixos adotados para este experimento, como tamanho da linha e associatividade da memória *cache*.

Tabela 3. Descrição da modelagem da memória *cache* L2 do primeiro experimento.

Tamanho por Banco	1 MB	2 MB	4 MB	8 MB	16 MB	32 MB
Tamanho de Linha	64 B	64 B	64 B	64 B	64 B	64 B
Associatividade	8 Ways	8 Ways	8 Ways	8 Ways	8 Ways	8 Ways
Latência de Acesso	1,77 ns	2,16 ns	2,70 ns	3,78 ns	4,82 ns	7,02 ns
Ciclos de Latência	4 Ciclos	5 Ciclos	6 Ciclos	8 Ciclos	10 Ciclos	15 Ciclos
Energia Dinâmica (nJ)	0,690	0,648	1,008	1,417	2,387	3,602
Potência Estática (W)	0,361	0,648	1,344	2,537	5,473	10,238
Área Ocupada (mm^2)	7,307	10,958	24,528	38,850	81,773	183,353
Portas de Leitura/Escrita	1	1	1	1	1	1

Este primeiro experimento tem como objetivo avaliar qual o comportamento do sistema ao agregar mais núcleos compartilhando o mesmo banco de memória *cache*, onde o tamanho total dessa memória é mantido. Para que este experimento represente as condições mais próximas da real, o parâmetros das memórias *cache* em todos experimentos foram obtidas pelo Cacti [Thoziyoor et al. 2008]. A Tabela 3 apresenta os níveis dos fatores modelados nesse primeiro experimento. A partir deste, outros testes foram planejados, a fim de avaliar outros fatores como tamanho da memória cache, tamanho do bloco, associatividade, níveis na hierarquia e influência do número de portas de acesso às memórias cache, investigando a correlação entre essas arquiteturas de memória *cache* e os diversos tipos de aplicações da carga de trabalho.

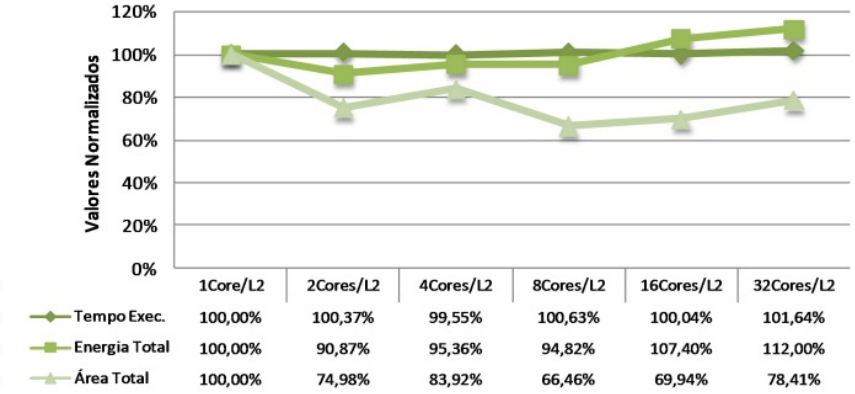


Figura 7. Sumário de desempenho, consumo de energia e ocupação de área do primeiro experimento.

A Figura 7 apresenta um sumário com valores de desempenho, consumo de potência e ocupação de área para esse experimento. Podemos notar, pelo gráfico, que a organização mais vantajosa em termos de desempenho é a organização 4Cores/L2 onde

obtemos ganho de 0,45% no desempenho, ao mesmo tempo que obtém-se redução de 4,64% na potência consumida e 16,08% na ocupação de área.

No entanto, o menor consumo de potência foi obtido na organização 2Cores/L2, com redução em 9,13%. Já a menor ocupação de área fica na configuração 8Cores/L2, reduzindo em 30,06% sem perda significativa de desempenho.

Os resultados finais obtidos ao término dessa pesquisa [Alves 2009] mostram a importância da integração entre os projetos de arquitetura de memória *cache* e o projeto físico da memória, a fim de obter o melhor equilíbrio entre tempo de acesso à memória *cache* e redução de faltas de dados. Nota-se nos resultados, dentro do espaço de projeto avaliado, que devido às limitações físicas e de desempenho, as organizações 1Core/L2 e 2Cores/L2, com tamanho total igual a 32 MB (bancos de 2 MB compartilhados), tamanho de linha igual a 128 *bytes*, representam uma boa escolha de implementação física em sistemas de propósito geral, obtendo um bom desempenho em todas aplicações avaliadas sem grandes sobrecustos de ocupação de área e consumo de energia.

Além disso, mostra-se que, para as atuais e futuras tecnologias de integração, as tradicionais técnicas de ganho de desempenho obtidas com modificações na memória *cache*, como aumento do tamanho das memórias, incremento da associatividade, maiores tamanhos da linha, *etc*., não devem apresentar ganhos reais de desempenho caso o acréscimo de latência gerado por essas técnicas não seja reduzido, a fim de equilibrar entre a redução na taxa de faltas de dados e o tempo de acesso aos dados.

5. Observações Finais e Perspectivas para o Futuro

Este artigo apresentou uma visão do GPPD sobre a introdução do paralelismo como fator chave na resolução de alguns desafios na pesquisa brasileira em computação, especialmente na construção de sistemas escaláveis para processamento distribuído e de alto desempenho para grandes volumes de dados. Relacionamos isso à elaboração de sistemas computacionais ecologicamente planejados (*green computing*) ao mostrar que os trabalhos publicados (tanto pelo grupo quanto por outros grupos) oferecem ferramentas importantes para conciliar o ganho de desempenho e o uso eficiente de recursos com a manutenção do consumo de energia elétrica próximo do mínimo.

Dado um fornecimento de potência elétrica constante por parte da fonte de alimentação, apresentamos duas metas chave para o aproveitamento eficiente da energia elétrica disponível: (1) a diminuição do tempo total de execução, obtido pela melhora do *speedup*; e (2) a redução do consumo de energia, obtido através do desligamento seletivo de processadores sub-aproveitados. A meta (1) é tratada através da aplicação de técnicas específicas relacionadas à programação paralela, onde abordamos o escalonamento de tarefas paralelas em processadores ociosos e o controle de granularidade de tarefas para a minimização do número total de processadores necessários à computação. A meta (2) é tratada através do uso eficiente da hierarquia de memórias *cache* em cada processador. Mostramos que as técnicas apresentadas, além de endereçar o problema do processamento eficiente e escalável de dados em larga escala, contribuem para a redução no desperdício de energia elétrica durante a computação.

Um aspecto importante, não abordado, que acreditamos ganhará destaque no futuro da computação verde é o uso eficiente da memória. Além da execução de fluxos de

execução sobre largos volumes de dados, necessita-se de um trabalho sólido que se preocupe com o armazenamento desses dados, principalmente frente a iniciativas como *cloud computing* e aos desafios #1 e #5 propostos pela SBC. Nesse sentido, desenvolvemos trabalhos que levam em conta a utilização de *clusters* de grande escala, que comumente apresentam recursos ociosos – nós de execução não utilizados. Estes recursos podem ser utilizados, mesmo que de forma temporária, pelo sistema de armazenamento como servidores de dados, de forma a aumentar o desempenho de I/O das aplicações em execução. Esta estratégia é implementada no sistema de arquivos dNFSp [Kassick et al. 2005] – um sistema de arquivos paralelo compatível com o protocolo NFS padrão. Este trabalho está alinhado com iniciativas semelhantes de outros grupos na área de *green computing*, como, *e.g.*, [Manzanares et al. 2008], que propõe a diminuição de consumo enegético em *data centers* por meio do armazenamento de resultados a consultas frequentes ao sistema de arquivos ("*prefetching*").

É importante destacar que as métricas (redução do consumo total de energia e número de recursos ativos) consideradas pelo nosso trabalho e em trabalhos na mesma linha (*e.g.*, [Binder and Suri 2009]) não constituem o único fator de otimização em Computação Verde. Conforme aponta [Hsu et al. 2005], existem outros fatores que também devem ser considerados, como o custo total de manutenção de um sistema verde. Na nossa visão, essas métricas, pouco presentes nos artigos atuais, passarão a ganhar mais relevância nos próximos anos.

Em resumo, acreditamos que o paralelismo, quando aliado ao uso otimizado de recursos ociosos, além de introduzir mecanismos eficientes para a realização de *green computing*, é uma solução escalável e de compromisso entre a tecnologia disponível atualmente e os avanços tecnológicos pontuais em direção à resolução dos desafios #1 e #5 na lista de desafios da SBC. Nessa direção, esperamos apresentar resultados importantes nos próximos anos.

Referências

Alves, M. A. Z. (2009). Avaliação do compartilhamento das memórias *Cache* no desempenho de arquiteturas *Multi-Core*. *PPGC / UFRGS -Programa de Pós-Graduação em Computação da Universidade Federal do Rio Grande do Sul*.

Alves, M. A. Z., Freitas, H. C., and Navaux, P. O. A. (2009). Investigation of shared l2 cache on many-core processors. In *Proceedings Workshop on Many-Core*, pages 21–30, Berlin. VDE Verlag GMBH.

Asanovic, K., Bodik, R., Demmel, J., Keaveny, T., Keutzer, K., Kubiatowicz, J., Morgan, N., Patterson, D., Sen, K., Wawrzynek, J., Wessel, D., and Yelick, K. (2009). A view of the parallel computing landscape. *Communications of the ACM*, 52(10).

Balaji, P., Buntinas, D., Goodwell, D., Gropp, W., and Thakur, R. (2008). Toward efficient support for multithreaded mpi communication. In *Proc. of The 15th European PVM/MPI Users' Group Conference, EuroPVM/MPI 2008*, pages 120–129, Dublin, IRL. Springer-Verlag.

Bardine, A., Comparetti, M., Foglia, P., Gabrielli, G., Prete, C. A., and Stenström andm, P. (2008). Leveraging data promotion for low power d-nuca caches. In *Digital System*

Design Architectures, Methods and Tools, 2008. DSD '08. 11th EUROMICRO Conference on, pages 307–316.

Binder, W. and Suri, N. (2009). Green computing: Energy consumption optimized service hosting. In *SOFSEM '09: Proceedings of the 35th Conference on Current Trends in Theory and Practice of Computer Science*, pages 117–128, Berlin, Heidelberg. Springer-Verlag.

Bjerregaard, T. and Mahadevan, S. (2006). A survey of research and practices of network-on-chip. *ACM Computing Surveys*, 38:1–51.

Cera, M. C., Maillard, N., and Navaux, P. O. A. (2007). A Centralized and On-line Scheduling Solution to Dynamic MPI Programs. *V Workshop de Processamento Paralelo e Distribuído (WSPPD 2007)*, pages 25–30.

De Micheli, G. and Benini, L. (2006). *Networks on Chips: Technology and Tools*. Morgan Kaufmann.

Dongarra, J., Foster, I., Fox, G., Gropp, W., Kennedy, K., Torczon, L., and White, A. (2003). *Sourcebook of Parallel Computing*. Morgan Kaufmann Publishers Inc., San Francisco, USA.

Feng, W. (2003). Green Destiny + mpiBLAST = Bioinfomagic. In *10th International Conference on Parallel Computing (ParCo)*.

Feng, W., Feng, X., and Ge, R. (2008). Green supercomputing comes of age. *IT Professional*, 10:17–23.

Flynn, M. J. (1972). Some computer organization and their effectiveness. In *IEEE Transactions on Computers*, volume C-21.

Foster, I. (1995). *Designing and Building Paralllel Programs*. Addison-Wesley.

Freitas, H. C., Colombo, D. M., Kastensmidt, F. L., and Navaux, P. O. A. (2007). Evaluating network-on-chip for homogeneous embedded multiprocessors in fpgas. In *Proceedings ISCAS: Int. Symp. on Circuits and Systems*, pages 3776–3779.

Hsu, C.-H., Feng, W., and Archuleta, J. S. (2005). Towards efficient supercomputing: A quest for the right metric. *Parallel and Distributed Processing Symposium, International*, 12:230a.

Jin, H., Frumkin, M., and Yan, J. (1999). The openmp implementation of nas parallel benchmarks and its performance. In *Technical Report: NAS-99-011*.

Kassick, R., Machado, C., Hermann, E., Ávila, R., Navaux, P., and Denneulin, Y. (2005). Evaluating the performance of the dnfsp file system. In *Proc. of the 5th IEEE International Symposium on Cluster Computing and the Grid, CCGrid*, Cardiff, UK. Los Alamitos, IEEE Computer Society Press. CD-ROM Proceedings, ISBN 0-7803-9075-X.

Kim, C., Burger, D., and Keckler, S. Q. (2002). An adaptive, non-uniform cache structure for wire-delay dominated on-chip-caches. In *Proceedings Int. Conf. on Architectural Support for Programming Languages and Operating Systems*, pages 211–222. IEEE.

Lima, J. and Maillard, N. (2009). Online mapping of mpi-2 dynamic tasks to processes and threads. *International Journal of High Performance Systems Architecture*, 2(2):81–89.

Manzanares, A., Bellam, K., and Qin, X. (2008). A prefetching scheme for energy conservation in parallel disk systems. In *Proceedings of The 22nd IEEE International Symposium on Parallel and Distributed Processing, IPDPS 2008*, pages 1–5.

Marino, M. D. (2006). 32-core cmp with multi-sliced l2: 2 and 4 cores sharing a l2 slice. In *Proceedings SBAC-PAD: Int. Symp. on Computer Architecture and High Performance Computing*, pages 141–150. IEEE.

Mór, S. D. K. and Maillard, N. B. (2009). Melhorando o desempenho de algoritmos do tipo branch & bound em mpi via escalonador com roubo aleatório de tarefas. In SBC, editor, *Anais do X Simpósio em Sistemas Computacionais, WSCAD-SSC 2009*, pages 11–18, São Paulo, BRA. SBC.

Navaux, P. O. A. and DeRose, C. A. F. (2003). *Arquiteturas Paralelas*. Number 15 in Série Livros Didáticos. Editora Sagra Luzzato, 1 edition.

Olukotun, K., Nayfeh, B. A., Hammond, L., Wilson, K., and Chang, K. (1996). The case for a single-chip multiprocessor. In *Proceedings ASPLOS: Int. Symp. on Architectural Support for Programming Languages and Operating Systems*, pages 2–11. IEEE.

Pezzi, G. P., Cera, M., Mathias, E., and Maillard, N. (2007). On-line scheduling of mpi-2 programs with hierarchical work stealing. *Computer Architecture and High Performance Computing, 2007. SBAC-PAD 2007. 19th International Symposium on*, pages 247–254.

Smith, J. E. and Sohi, G. S. (1995). The microarchitecture of superscalar processors. *IEEE*, 83(12):1609–1624.

Thoziyoor, S., Ahn, J. H., Monchiero, M., Brockman, J. B., and Jouppi, N. P. (2008). A comprehensive memory modeling tool and its application to the design and analysis of future memory hierarchies. In *Proceedings ISCA: Int. Symp. on Computer Architecture*, pages 51–62.

Torres, J., Carrera, D., Hogan, K., Gavaldà, R., Beltran, V., and Poggi, N. (2008). Reducing wasted resources to help achieve green data centers. In *Proceedings of The 22nd IEEE International Symposium on Parallel and Distributed Processing, IPDPS 2008*, pages 1–8.

Vykoukal, J., Wolf, M., and Beck, R. (2009). Does green it matter? analysis of the relationship between green it and grid technology from a resource-based view perspective. In *PACIS 2009 Proceedings*, page paper 51.

Wang, D. (2007). Meeting green computing challenges. In *International Symposium on High Density packaging and Microsystem Integration*, pages 1–4.