Avaliação do Consumo Energético em Arquiteturas Multi-Core com Memória Cache Compartilhada

Matheus A. Souza[1], **Henrique C. Freitas**[1],
Marco A. Z. Alves[2], **Philippe O. A. Navaux**[2]

[1]Grupo de Arquitetura de Computadores e Processamento Paralelo (CArT)
Departamento de Ciência da Computação
Pontifícia Universidade Católica de Minas Gerais (PUC Minas)
Belo Horizonte, Brasil

[2]Grupo de Processamento Paralelo e Distribuído (GPPD)
Instituto de Informática
Universidade Federal do Rio Grande do Sul (UFRGS)
Porto Alegre, Brasil

matheus.alcantara@sga.pucminas.br, cota@pucminas.br

{mazalves,navaux}@inf.ufrgs.br

Abstract. *A design challenge for multi-core processors is to obtain the best possible level of energy efficiency. This paper presents results concerning the energy consumption of a simulated chip-multiprocessor architecture (CMP) with different shared L2 cache models under different multi-threaded workloads. The decrease in cache size and its distribution in groups of processing cores made possible an overall reduction in power consumption up to 18.10%, and 70.51% at the memory access, with decrease in performance up to 57.13% and increase in L2 cache misses up to 93.66%. In general, the version with 32 L2 private cache banks consumed up to 53.27% less energy.*

Resumo. *Um desafio no projeto de processadores multi-core é obter o melhor nível de eficiência energética possível. Este artigo apresenta resultados relativos ao consumo de energia de uma arquitetura de chip multiprocessado (CMP) simulada, com diferentes modos de compartilhamento de cache L2 sob cargas de trabalho multi-threaded variadas. A diminuição do tamanho da cache e sua distribuição em grupos de núcleos permitiram uma redução de potência consumida de até 18,10% no total e de 70,51% no acesso a memória, porém, com queda de desempenho de até 57,13% e aumento da taxa de faltas na cache L2 de até 93,66%. Em linhas gerais, a versão com 32 bancos de cache L2 privadas consumiu até 53,27% menos energia.*

1. Introdução

Diversas pesquisas vêm sendo realizadas para que seja possível o desenvolvimento de arquiteturas de computadores que tenham alta eficiência energética. Em *chips* multiprocessados (CMPs) com um grande número de núcleos, a intercomunicação entre as unidades de processamento torna-se mais um fator desafiante para a construção de arquiteturas com a eficiência energética desejada [Kin et al. 1997] [Hennessy and Patterson 2006] [Olukotun et al. 1996] [Sundararajan et al. 2013] [Kumar et al. 2005] [Wolf 2004].

Além do problema relacionado à interconexão entre os processadores, deve-se levar em consideração a organização de memória nos CMPs. A maioria desses CMPs é construída com diferentes organizações de memória *cache*, tipicamente com uma *cache* de último nível, ou *Last Level Cache* (LLC), compartilhada entre os núcleos do *chip*. Ainda que esse modelo de organização de *cache* apresente desempenho satisfatório em algumas arquiteturas [Nayfeh and Olukotun 1994], os dispositivos de memória contidos nos CMPs são responsáveis por grande parte do consumo de energia do sistema [Li et al. 2009]. Dessa forma, o problema abordado neste artigo pode ser sintetizado na seguinte pergunta: Como o tamanho e o modo de compartilhamento da LLC pode influenciar no consumo de energia?

Diante da questão apresentada, este trabalho visa apresentar um comparativo de arquiteturas de CMPs com diferentes modelos de compartilhamento da *cache* LLC, observando aspectos relativos ao consumo de energia e ao comportamento das memórias *cache* L2 (representando a LLC neste trabalho).

A contribuição principal deste artigo está no estudo de arquiteturas de *chips* multiprocessados com *cache* L2 compartilhada e na identificação da arquitetura que apresenta a melhor relação de consumo de energia por desempenho. Os resultados foram gerados usando o simulador de arquiteturas superescalares SESC [Renau et al. 2005] integrado ao Wattch [Brooks et al. 2000], usando como carga de trabalho, o conjunto de aplicações *multi-threaded* SPLASH-2 [Woo et al. 1995].

O restante deste trabalho está estruturado como segue. A Seção 2 apresenta alguns trabalhos de temas correlatos. Na Seção 3 constam as arquiteturas propostas e a metodologia estabelecida para simulação. Na Seção 4 está a discussão dos resultados obtidos nas simulações. A Seção 5 encerra o trabalho, com as conclusões e sugestões de trabalhos futuros.

2. Trabalhos Correlatos

Diversos trabalhos têm sido realizados para identificar alternativas de arquiteturas de processadores que promovam alta eficiência energética. Esta seção contém uma visão geral de alguns desses trabalhos.

Com foco no consumo de energia, em [Sundararajan et al. 2013] é apresentada uma arquitetura dinâmica que permite a reconfiguração do tamanho e da associatividade das memórias *cache* durante a execução. É feita uma comparação dessa arquitetura com outras arquiteturas reconfiguráveis similares, com capacidades similares de reconfiguração. Baseados nos resultados do trabalho, os autores afirmam que a arquitetura proposta apresenta melhoras na eficiência energética de 70%, em média, com queda máxima de desempenho menor que 2%.

Ainda relacionado ao tema de eficiência energética, [Kin et al. 1997] trata da quantidade significante de potência consumida pelas memórias *cache* em um processador, o que é justificado pelas características físicas desses componentes e pelo espaço que eles ocupam em um chip. A proposta do artigo é reduzir a potência consumida da memória *cache* através da adição de uma *cache* intermediária denominada *Filter Cache*, em contrapartida reduzindo o desempenho do processador. Em uma das simulações realizadas, foi alcançada uma redução de potência consumida de 58% com uma queda de desempenho de 21%.

O estudo realizado em [Alves et al. 2007] apresenta resultados de simulações em arquiteturas CMP com *cache* L2 compartilhada. O objetivo das simulações feitas é avaliar como essas arquiteturas se comportam ao processar cargas de trabalho com conjunto de dados contíguos e não contíguos em memória. Os resultados mostraram que, à medida que o agrupamento de processadores por *cache* L2 aumenta, há uma redução na taxa de faltas para leitura e aumento na taxa de faltas para escrita na referida *cache*.

Em [Alves et al. 2011], os autores avaliam limitações (*bottlenecks*) encontradas em arquiteturas CMP com memória *cache* L2 compartilhada, medindo o impacto dessas limitações no desempenho. De acordo com o trabalho, o uso de memória *cache* compartilhada pode provocar restrições na latência de acesso, potência consumida e na área ocupada pelos componentes no *chip*. Os resultados para simulações com 32 núcleos mostraram desempenho menor quando a memória *cache* L2 é compartilhada entre 2 ou 4 núcleos, o que é justificado pelo aumento da latência de acesso e contenção na *cache*.

O trabalho desenvolvido em [Youn et al. 2007] propõe uma organização de memória *cache* L2 diferenciada, que une as vantagens das *caches* privada e compartilhada. A proposta é que cada núcleo possua sua *cache* L2 privada, conectadas por um barramento compartilhado. Quando um bloco de *cache* tende a ser reutilizado, este é alocado em alguma *cache* L2 vizinha que contém blocos inválidos. Simulações em um *chip* multiprocessado mostram uma melhora de 14% na latência média de acesso à memória.

Em [Chtioui et al. 2012], é realizado um trabalho com foco em *Multi-Processor System-on-Chips* (MPSoCs) com o uso de modelos de programação em memória compartilhada. O artigo propõe um novo protocolo de coerência de *cache* com o objetivo de melhorar o desempenho do processador. Os resultados obtidos mostraram que o novo protocolo permitiu redução de 77% na taxa de faltas na memória *cache*, o que, segundo os autores, provocou redução de 20% no tempo de execução e de 34% na energia consumida.

Uma nova organização de memória *cache* é apresentada em [Garcia-Guirado et al. 2012]. Denominada DAPSCO (*Distance-Aware Partially Shared Cache Organization*), essa organização propõe uma política de mapeamento de endereços de memória, que encontra o melhor caminho para acesso aos bancos de memória, diminuindo o consumo de energia e o desempenho do sistema. Os testes em uma arquitetura CMP de 64 núcleos mostraram que essa política de mapeamento promove uma melhora de cerca de 13% no tempo de execução e no consumo de energia.

Desafios para computação em exaflop dependem de redução de consumo energético e do uso de processadores many-core. O trabalho proposto neste artigo é complementar aos artigos correlatos citados. São feitas simulações com cargas de trabalho variadas com agrupamentos de *cache* L2 similares a [Alves et al. 2007]. Entretanto, o tamanho da LLC aumenta à medida em que o número de núcleos por agrupamento também aumenta. Além disso, são usadas mais cargas de trabalho (seis ao todo) e os resultados obtidos apresentam informações sobre o consumo energético de cada arquitetura CMP simulada.

3. Metodologia

Nesta seção, é apresentada a proposta das simulações realizadas para coleta dos resultados, em termos de desempenho e consumo energético.

3.1. Arquitetura Simulada

As simulações propostas neste trabalho possuem como base a arquitetura do processador MIPS. Trata-se de uma arquitetura de 64 bits baseada em um conjunto de instruções *load-store*. A arquitetura MIPS oferece um modelo conveniente para estudos por ser completa e de fácil compreensão [Hennessy and Patterson 2006].

Foram simulados seis organizações diferentes da LLC. Todas as organizações possuem 32 núcleos em um *chip* multiprocessado e cada núcleo possui uma memória *cache* L1 privada para dados e uma para instruções. A diferença nos modelos de arquitetura, explicada a seguir, está na organização da memória *cache* L2, que neste trabalho é o último nível de *cache* (LLC).

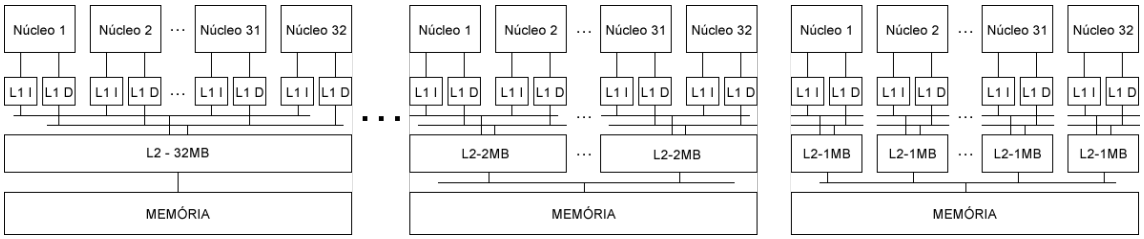

Figura 1. Organizações da *cache* L2 nas arquiteturas simuladas

Foram estabelecidos diferentes agrupamentos de memória *cache* L2. Esses agrupamentos variam de uma memória *cache* L2 compartilhada entre todos os núcleos do *chip* até uma por núcleo (privada). Em todos os casos, a memória *cache* L2 está dentro do *chip*. A Figura 1 apresenta os modelos simulados para 1 banco com tamanho de 32 MB compartilhado entre todos os 32 núcleos, 16 bancos com tamanho de 2 MB para cada 2 núcleos e 32 bancos com tamanho de 1 MB para cada núcleo, respectivamente.

As demais configurações do processador permanecem iguais em todos os casos simulados. A Tabela 1 apresenta as configurações a serem simuladas.

Os dados de latência de acesso foram obtidos com base na ferramenta CACTI versão 4.0 [Tarjan et al. 2006] que será apresentada na seção posterior. Para a *cache* L1 foi calculado o valor de 1 ciclo. Para a *cache* L2 foram calculados 2 ciclos quando não compartilhadas (1 núcleo apenas) ou compartilhadas entre 2 núcleos, 3 ciclos quando compartilhadas entre 4 ou 8 núcleos, 4 ciclos quando compartilhadas entre 16 núcleos, e 6 ciclos quando compartilhadas entre 32 núcleos.

De forma a facilitar a leitura dos resultados, definiu-se uma nomenclatura para identificação dos modelos de agrupamento simulados. O nome *M*cj*N*p será usado, onde M=$\{1,2,4,8,16,32\}$ representa o número de bancos ou conjuntos (cj) de memória *cache* L2 e N=$\{32,16,8,4,2,1\}$ representa o número de núcleos (p) que compartilham este banco e também o tamanho da memória *cache* L2, dado em *mega bytes*. A Tabela 2 apresenta um resumo dos modelos simulados.

3.2. Simuladores

Para simulação de desempenho da arquitetura proposta, foi escolhido o simulador SESC (*Super Escalar Simulator*) [Renau et al. 2005]. O SESC é um simulador de arquitetura de processadores que adota um modelo de simulação determinístico. Nele, é possível

Tabela 1. Configurações do Sistema Simulado

Componente	Parâmetro	Configuração
Processador	Quantidade de núcleos	32 núcleos MIPS
	Quantidade de estágios	5
	Frequência	1 Ghz
	Tecnologia de integração	32 nm
	Execução	Fora de ordem
	Fetch width	1 instrução
	Issue width (IPC máx.)	1 instrução
	Execução	Fora de ordem
	ROB	68 entradas
Cache L1	Tamanho da linha	64 B
	Tamanho - Dados (D)	32 kB
	Tamanho - Instruções (I)	32 kB
	Política de substituição	LRU
	Política de escrita (D)	Write Back
	Conjunto associativo (D)	4 vias
	Conjunto associativo (I)	2 vias
	Latência	1 ciclo
	Protocolo de coerência	MESI
Cache L2	Tamanho da Linha	64 B
	Política de substituição	LRU
	Política de escrita	Write Back
	Conjunto associativo	8 vias
	Latência	Ver Tabela 2
Memória Principal	Latência	200 ciclos

Tabela 2. Modelos de agrupamento

Nome	Bancos	Núcleos por banco	Tamanho da *cache* L2 por banco	Latência da *cache* L2
1cj32p	1	32	32 MB	6 ciclos
2cj16p	2	16	16 MB	4 ciclos
4cj8p	4	8	8 MB	3 ciclos
8cj4p	8	4	4 MB	3 ciclos
16cj2p	16	2	2 MB	2 ciclos
32cj1p	32	1	1 MB	2 ciclos

modelar tanto processadores *single-core* quanto *chips* multiprocessados. É possível modelar nesse simulador um *pipeline* superescalar fora de ordem com predição de desvio, memórias *cache*, barramentos e outros componentes.

Para obter resultados em termos de energia, usamos o Wattch [Brooks et al. 2000], que é uma ferramenta desenvolvida na Universidade de Princeton cujo propósito é permitir a análise da potência consumida em processadores no nível da arquitetura. Para cada unidade funcional do processador, é criado um modelo de potência dissipada, tomando como premissa uma simulação ciclo-a-ciclo orientada a eventos. O consumo de potência total é calculado com base no modelo criado para cada unidade e o número de acessos a elas.

O modelo de potência do Wattch é baseado em outros modelos de potência de processadores superescalares reais. Os componentes principais do processador são organizados em quatro categorias: estruturas de arranjo, memórias completamente as-

sociativas de conteúdo endereçável, lógica combinacional e fios, e *clocking*. Alguns parâmetros desses componentes são informados para o Wattch, que produz os resultados de potência com base em modelos matemáticos. O detalhamento completo dos modelos matemáticos, bem como dos parâmetros de componentes necessários, podem ser verificados em [Brooks et al. 2000].

A ferramenta CACTI [Tarjan et al. 2006] foi utilizada para obter parâmetros referentes às memórias cache. Mantida pela Hewlett-Packard, a ferramenta possibilita realizar estimativas de potência consumida em memórias *cache* no nível da arquitetura. O uso dessa ferramenta permite a análise da configuração de memória *cache* que melhor atende a determinados critérios de desempenho.

As ferramentas Wattch e CACTI foram acopladas no pacote de utilitários do simulador SESC, para que seja possível a simulação de processadores com resultados relativos ao consumo de energia e potência, bem como do desempenho. Esse conjunto de simuladores (SESC, CACTI e Wattch) estão validados pela comunidade científica, sendo usados no desenvolvimento de diversos trabalhos envolvendo simulação de arquiteturas de processadores [Staples 2011] [Ganeshpure et al. 2009] [Nogueira 2013] [Rawlins and Gordon-Ross 2012] [Weng 2012].

3.3. Cargas de Trabalho Utilizadas

De posse da arquitetura proposta e definido o simulador, foram escolhidas cargas de trabalho variadas do SPLASH-2 [Woo et al. 1995]. Este conjunto de cargas de trabalho reais é formado por 11 aplicações paralelas com propósitos diversos, principalmente no domínio de computação de alto desempenho. O SPLASH-2 é amplamente utilizado em simulações e avaliação de sistemas de memória.

Tabela 3. Parâmetros das aplicações

#	Aplicação	Descrição	Parâmetro	Valor
1	LU-CON (Dados contíguos)	Algoritmo para fatorização de uma matriz densa na forma do produto de uma matriz triangular inferior e outra superior	Matriz	1024 x 1024
2	LU-NON (Dados não contíguos)		Blocos	64 x 64
3	OCEAN-CON (Dados contíguos)	Aplicações que fazem uma representação de movimentos de larga escala do oceano, baseado em redemoinhos e correntes de limite	Grade	258 x 258
4	OCEAN-NON (Dados não contíguos)			
5	WATER N-SQUARED	Aplicações que avaliam forças e potenciais moleculares que ocorrem ao longo do tempo em um sistema de moléculas de água	Moléculas	4096
6	WATER SPATIAL			

Apesar do número de aplicações disponibilizadas pelo SPLASH-2, algumas não foram utilizadas neste trabalho. Foram escolhidas as aplicações que apresentaram em [Bienia et al. 2008] as maiores taxas de escrita em memória compartilhada, permitindo assim a avaliação dos componentes em foco neste trabalho (LLC).

As aplicações selecionadas foram: *LU* (dados contíguos e não contíguos), *Ocean* (dados contíguos e não contíguos), *Water N-squared* e *Water Spatial*, sendo compiladas para uso no simulador SESC. Os parâmetros para cada aplicação podem ser verificados na Tabela 3.

4. Avaliação de Resultados

Nesta seção, são apresentados os resultados obtidos nas simulações das arquiteturas propostas para as 6 aplicações selecionadas.

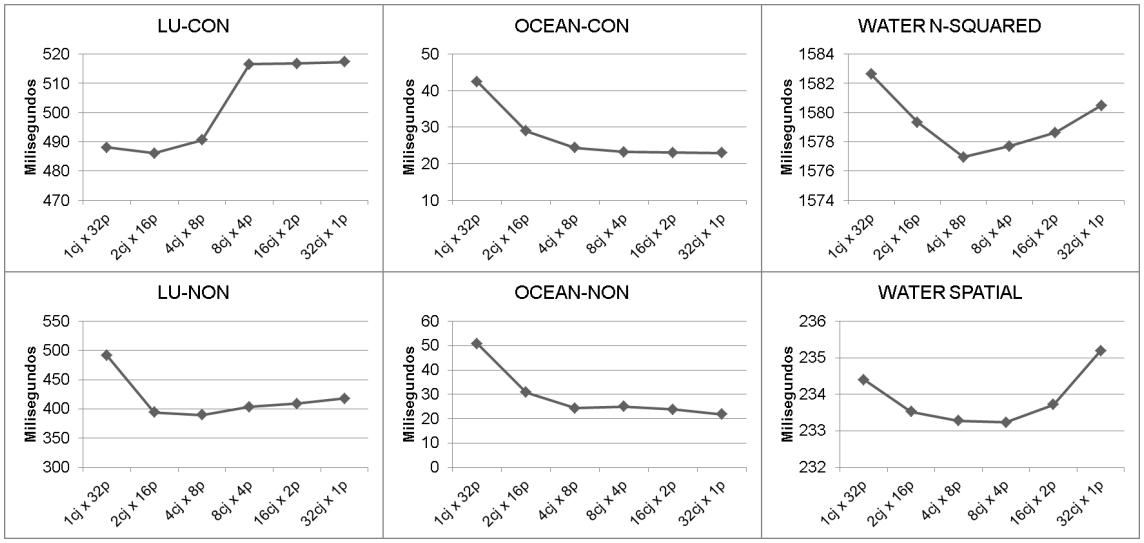

Figura 2. Tempo de execução (milisegundos)

A Figura 2 apresenta o gráfico com o tempo de execução das aplicações selecionadas. Nota-se que houve uma diferença entre o pior e o melhor desempenho de 6,04% para a aplicação *LU* com dados contíguos e de 20,68% para dados não contíguos. Já para a aplicação *Ocean*, houve uma diferença entre o pior e o melhor desempenho de 45,86% para dados contíguos e de 57,13% para dados não contíguos. Por fim, a aplicação *Water* apresentou as menores variações de desempenho, apresentando diferença entre o pior e melhor valor de 0,36% para *Water N-squared* e de 0,84% para *Water Spatial*.

As arquiteturas que apresentaram melhor desempenho foram a 2cj16p para a aplicação *LU* com dados contíguos, a 4cj8p para a aplicação *LU* com dados não contíguos e para *Water N-squared*, a 8cj4p para a aplicação *Water Spatial* e a 32cj1p para as duas aplicações *Ocean*.

Em contrapartida, os piores desempenhos foram apresentados pela arquitetura 1cj32p nas aplicações *LU* com dados não contíguos, *Ocean* em ambos os casos e para *Water N-squared*, enquanto a arquitetura 32cj1p cada apresentou o pior desempenho para a aplicação *LU* com dados contíguos e para *Water Spatial*.

É apresentado na Figura 3 o gráfico com a potência total dissipada pelo *chip* em cada aplicação e arquitetura simulada. Por este gráfico, percebemos que tipicamente os agrupamentos e tamanhos de memória *cache* L2 menores consomem menos potência. Isso se explica pelo fato de que uma LLC menor possui um circuito menor, o que permite um menor tempo de acesso aos dados.

Contudo, características das cargas de trabalho (aplicações), assim como o efeito do compartilhamento de dados, entre outras variáveis, podem resultar em outro comportamento. Por exemplo, a aplicação *LU* com dados não contíguos na arquitetura 1cj32p apresentou melhor resultado do que na arquitetura 2cj16p. Comportamento semelhante

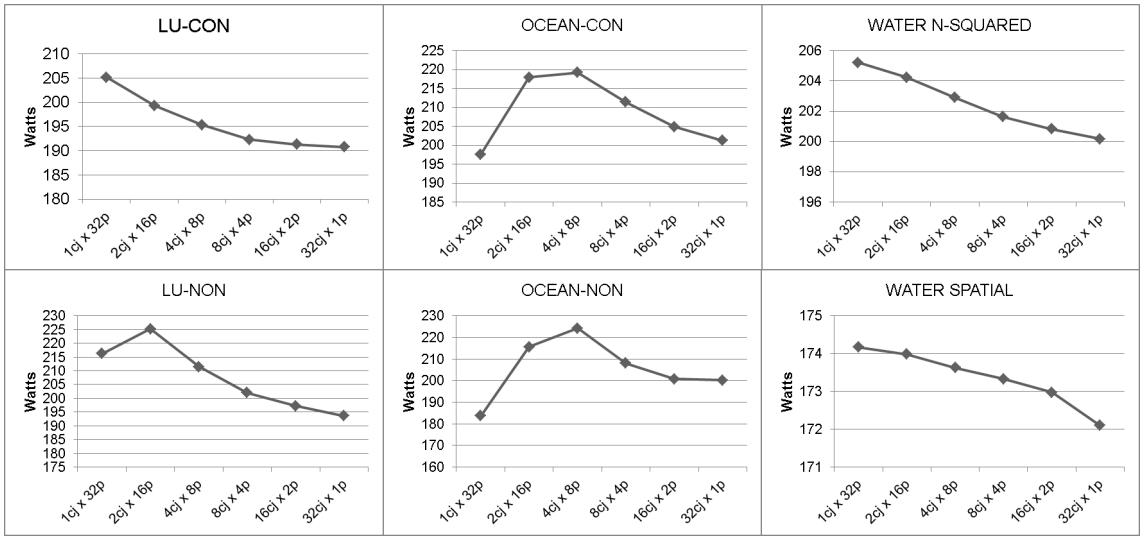

Figura 3. Potência consumida (Watts)

foi observado na aplicação *Ocean*, para a qual a arquitetura 1cj32p apresentou melhor resultado do que as demais no que se refere a potência consumida.

As diferenças entre o menor e maior consumo de potência foram de 7,03% para a aplicação *LU* com dados contíguos e de 14,00% para não contíguos, 9,86% para a aplicação *Ocean* com dados contíguos e 18,10% para dados não contíguos, 2,47% para *Water N-squared* e 1,19% para a aplicação *Water Spatial*.

Os melhores resultados, exceto os para a aplicação *Ocean* relatados anteriormente, foram obtidos na arquitetura 32cj1p. Já a arquitetura 1cj32p apresentou os piores resultados (aplicações *Water* e aplicação *LU* com dados contíguos), seguida pela 4cj8p na aplicação *Ocean* e por fim, na aplicação *LU* com dados não contíguos, a 2cj16p.

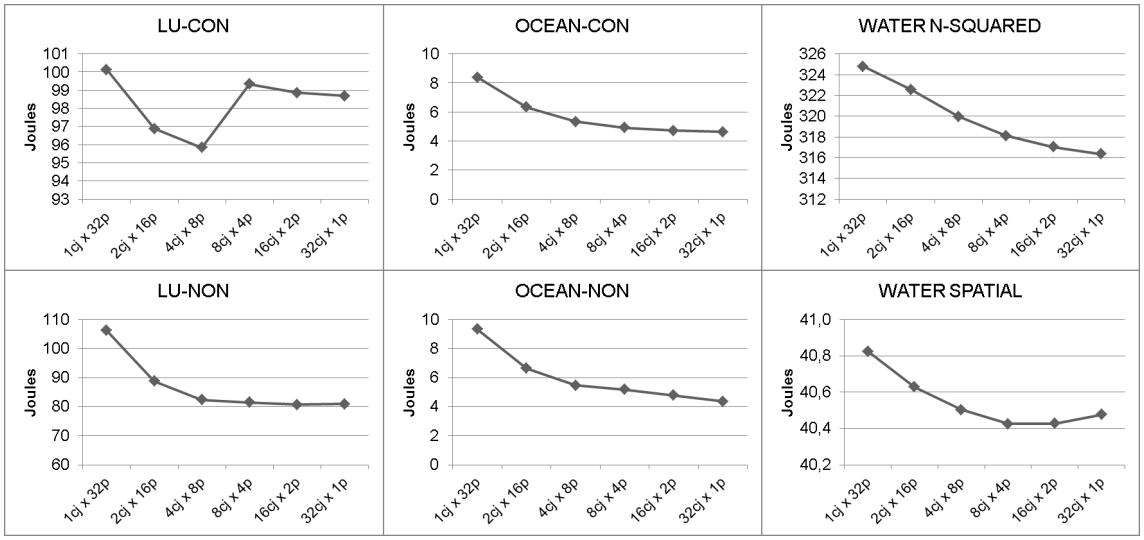

Figura 4. Energia total consumida (Joules)

Avaliando-se a energia consumida, através do gráfico apresentado na Figura 4, é constatado que para todas as aplicações, o pior resultado é para a arquitetura 1cj32p.

As aplicações *Ocean* com dados contíguos, não contíguos, e *Water N-squared* apresentaram os menores consumos de energia na arquitetura 32cj1p, com melhorias de 44,88%, 53,27% e 2,60%, respectivamente. As aplicações *LU* com dados não contíguos e *Water Spatial* apresentaram melhores resultados na arquitetura 16cj2p, com melhorias no consumo de energia de 24,02% e 0,98%, respectivamente. Por fim, a aplicação *LU* com dados contíguos teve melhores resultados de consumo de energia com a arquitetura 4cj8p, apresentando ganhos de 4,30% em relação ao pior resultado.

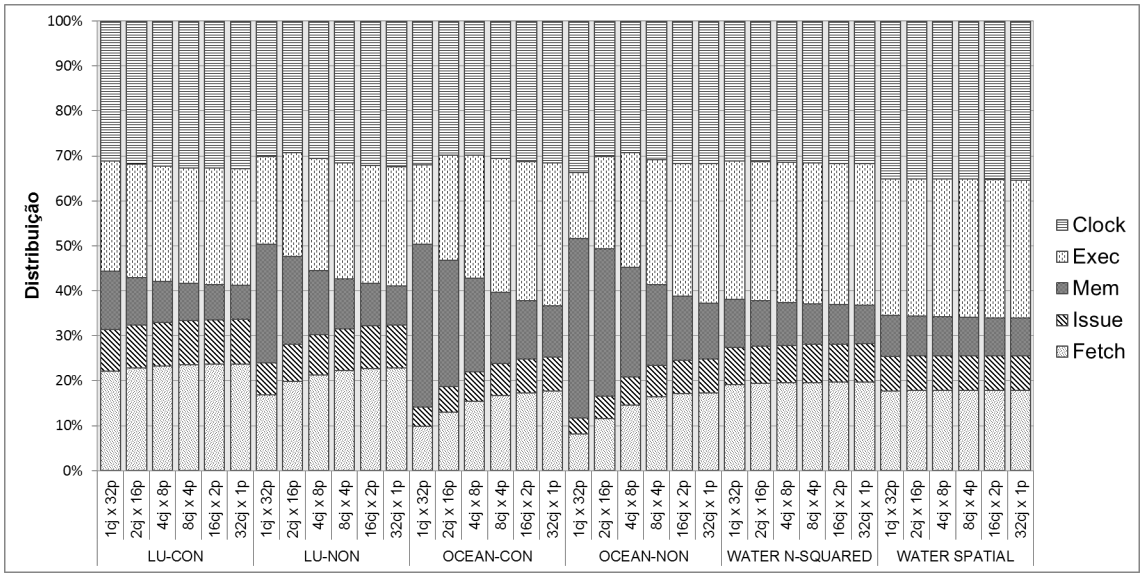

Figura 5. Distribuição da potência consumida no processador

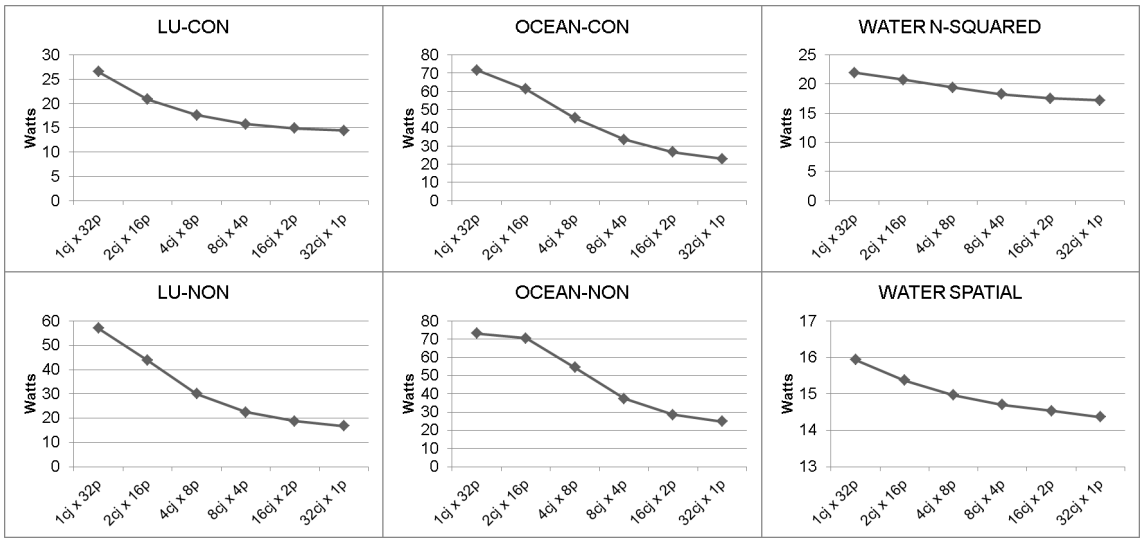

Figura 6. Potência consumida no acesso à memória (Watts)

Uma análise específica da potência consumida pela memória pode ser feita com a leitura dos gráficos apresentados na Figura 5 e na Figura 6. No gráfico da Figura 5, é apresentada a distribuição do consumo de potência no *pipeline* do processador. É possível notar, para todas as aplicações, uma queda gradativa da parcela de potência consumida no

acesso à memória, à medida em que o tamanho do banco de memória *cache* L2 é reduzido, diminuindo também o número de processadores agrupados em um único conjunto.

De forma complementar, o gráfico da Figura 6 permite constatar que, avaliando somente o acesso à memória, a arquitetura 1cj32p apresenta o pior consumo de potência em todas as aplicações simuladas. Já a arquitetura 32cj1p apresenta o melhor resultado, sendo o menor consumo de potência.

A aplicação *LU* com dados não contíguos apresentou o melhor percentual de redução no consumo de energia no acesso à memória (70,51%), seguido pela aplicação *Ocean* (67,95% para dados contíguos e 66,14% para dados não contíguos). A aplicação *LU* com dados contíguos apresentou redução de 45,75% enquanto as aplicações *Water N-squared* e *Water Spatial* apresentaram redução de 21,77% e 9,89% respectivamente.

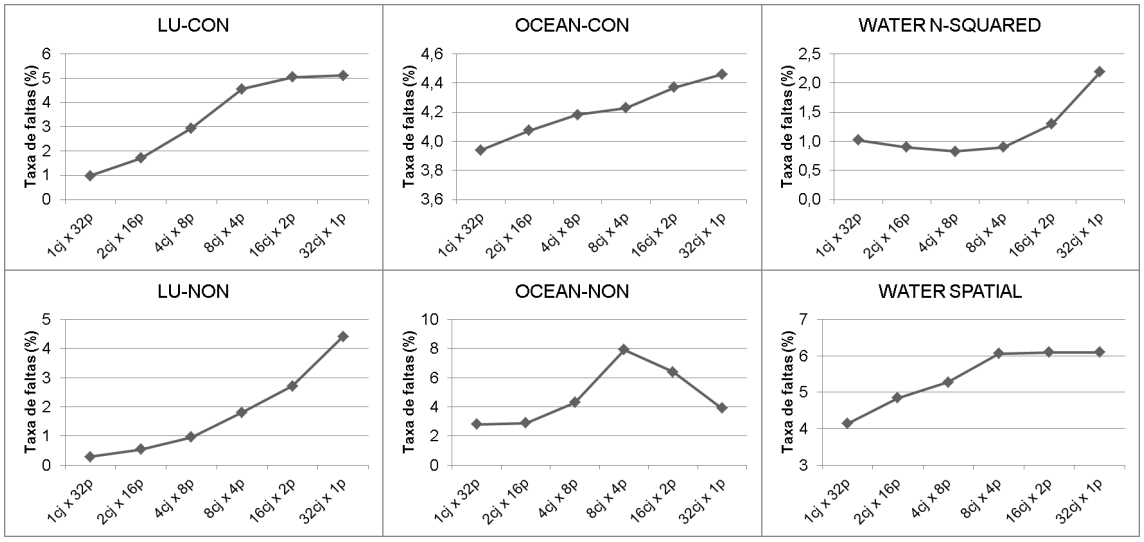

Figura 7. Taxas de faltas na memória *cache* L2 (%)

Como já discutido anteriormente, a ocorrência de *cache miss*, ou falta, durante o processamento influi diretamente no consumo de energia de um processador. O gráfico da Figura 7 ilustra a média das taxas de falta de cada agrupamento de memória *cache* L2, para cada simulação realizada.

É possível perceber pelo gráfico um comportamento frequente, mas que não ocorre em todos os casos, em que a taxa de faltas aumenta à medida em que o número de bancos também aumenta. Esse comportamento se justifica pela diminuição do tamanho da memória *cache* L2 em cada agrupamento. Além disso, na hipótese de uma situação em que um dado compartilhado é usado por dois processadores, e esses processadores não possuem uma *cache* compartilhada entre si, o dado é replicado em diferentes *caches*, o que não é desejável. Entretanto, se faz necessário entender detalhadamente o fluxo das *threads* da aplicação em execução para definir-se um padrão de comportamento.

Nota-se também uma exceção na aplicação *Ocean* com dados não contíguos, em que a partir da arquitetura de 8cj4p, a taxa média de faltas diminui com o aumento do número de bancos. Esse comportamento foi dado como anômalo também em [Alves et al. 2007], em que o autor sugere o particionamento e localidade dos dados da aplicação como causadores deste resultado.

Foi constatado também, para as aplicações *LU* e *Water Spatial*, que não há homogeneidade da taxa de faltas em todos os bancos, sendo tipicamente o primeiro banco de memória *cache* L2 o que apresenta a maior taxa de faltas, com os demais bancos apresentando taxas menores. Essa heterogeneidade da taxa de faltas nos bancos de memória reforça a necessidade de estudo detalhado do comportamento das *threads* da aplicação.

Através dos dados coletados e relatados, percebemos que a variação do compartilhamento da memória *cache* L2, alterando o agrupamento de processadores e o tamanho da memória em um agrupamento (sem alteração do tamanho total), interfere no desempenho e principalmente no consumo de potência. O uso de vários agrupamentos de memória com um número menor de processadores (32 bancos com 1 processador) claramente diminuiu o consumo de potência no acesso à memória. Entretanto, é necessário levar em consideração as características da aplicação executada.

Em geral, constatou-se também que a diminuição do tamanho da LLC resulta no aumento da taxa de faltas. Era esperado que o aumento da taxa de faltas provocasse o aumento do consumo energético total, o que não é um acontecimento padrão. O que ocorre é que a diminuição da LLC, que significa um circuito menor e, consequentemente, um menor tempo de acesso aos dados, causa a redução do consumo energético. Dada uma carga de trabalho, o consumo de potência por acesso é menor, resultando em um consumo de energia menor. Apesar do maior número de cache miss, o maior tempo de espera pelo acesso aos dados não é suficiente para aumentar o consumo de energia, quando há redução do tamanho da LLC.

5. Conclusões

O uso de *chips* multiprocessados (CMPs) com um grande número de núcleos já é uma realidade na construção de processadores de propósito geral. Nesse sentido, a preocupação com a arquitetura das memórias *cache* também é maior, de forma a acompanhar o desempenho e características do *chip* multiprocessado.

Outra preocupação dos projetistas de CMPs é obter o maior desempenho e o menor consumo de potência possível no uso dos *chips* multiprocessados. As memórias *cache* L2 influenciam diretamente no consumo de potência de um CMP, reforçando ainda mais a importância do estudo de melhores arquiteturas desses dispositivos.

Este trabalho apresentou um comparativo de arquiteturas de CMPs com diferentes modos de compartilhamento da *cache* L2, observando aspectos relativos ao consumo de potência e ao comportamento dessas memórias. Com os resultados obtidos, verificou-se que a variação do modelo de memória *cache* L2, no que se refere ao agrupamento de processadores e tamanho do agrupamento, influencia diretamente no consumo de potência e desempenho do CMP. Entretanto, não é possível definir genericamente um padrão de comportamento e uma melhor arquitetura avaliando somente a memória. As cargas de trabalho das aplicações paralelas executadas também devem ser avaliadas e observadas detalhadamente.

Como trabalhos futuros, sugere-se a avaliação de aspectos relativos ao consumo de energia em arquiteturas *Many Core* (e.g. *Networks-on-chip*, *Graphics Processing Units*) e também modelos de arquiteturas reconfiguráveis com foco em eficiência energética. Outros trabalhos podem avaliar qual o impacto provocado pela alteração no número de

bancos de LLC em relação ao protocolo de coerência de *cache*, bem como na contenção existente no acesso à *cache* pelos vários núcleos.

6. Agradecimentos

Os autores agradecem ao CNPq, CAPES, FAPERGS e FAPEMIG pelo suporte parcial na realização deste trabalho.

Referências

Alves, M. A., Freitas, H. C., Wagner, F. R., and Navaux, P. O. (2007). Influência do compartilhamento de cache l2 em um chip multiprocessado sob cargas de trabalho com conjuntos de dados contíguos e não contíguos. *VIII Workshop em Sistemas Computacionais de Alto Desempenho*, pages 27–34.

Alves, M. A. Z., Freitas, H. C., and Navaux, P. O. A. (2011). High latency and contention on shared l2-cache for many-core architectures. *Parallel Processing Letters*, 21(1):85–106.

Bienia, C., Kumar, S., and Li, K. (2008). Parsec vs. splash-2: A quantitative comparison of two multithreaded benchmark suites on chip-multiprocessors. In *Workload Characterization, 2008. IISWC 2008. IEEE International Symposium on*, pages 47–56.

Brooks, D., Tiwari, V., and Martonosi, M. (2000). Wattch: a framework for architectural-level power analysis and optimizations. In *Computer Architecture, 2000. Proceedings of the 27th International Symposium on*, pages 83–94.

Chtioui, H., Lamih, S. N., Ben-Atitallah, R., M.Zahran, Dekeyser, J., and Abid, M. (2012). Article: A dynamic hybrid cache coherency protocol for shared-memory mpsoc architectures. *International Journal of Computer Applications*, 47(3):45–50. Published by Foundation of Computer Science, New York, USA.

Ganeshpure, K. P., Polian, I., Kundu, S., and Becker, B. (2009). Reducing temperature variability by routing heat pipes. In *Proceedings of the 19th ACM Great Lakes symposium on VLSI*, pages 63–68. ACM.

Garcia-Guirado, A., Pascual, R. F., Ros, A., and 0001, J. M. G. (2012). Dapsco: Distance-aware partially shared cache organization. *TACO*, 8(4):25.

Hennessy, J. L. and Patterson, D. A. (2006). *Computer Architecture, Fourth Edition: A Quantitative Approach*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Kin, J., Gupta, M., and Mangione-Smith, W. (1997). The filter cache: an energy efficient memory structure. In *Microarchitecture, 1997. Proceedings., Thirtieth Annual IEEE/ACM International Symposium on*, pages 184–193.

Kumar, R., Zyuban, V., and Tullsen, D. M. (2005). Interconnections in multi-core architectures: Understanding mechanisms, overheads and scaling. *SIGARCH Comput. Archit. News*, 33(2):408–419.

Li, S., Ahn, J. H., Strong, R. D., Brockman, J. B., Tullsen, D. M., and Jouppi, N. P. (2009). Mcpat: An integrated power, area, and timing modeling framework for multicore and manycore architectures. In *Proceedings of the 42Nd Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO 42, pages 469–480, New York, NY, USA. ACM.

Nayfeh, B. and Olukotun, K. (1994). Exploring the design space for a shared-cache multiprocessor. In *Computer Architecture, 1994., Proceedings the 21st Annual International Symposium on*, pages 166–175.

Nogueira, L. O. (2013). *New Hardware Support for Transactional Memory and Parallel Debugging in Multicore Processors*. PhD thesis, Universidade de Santiago de Compostela.

Olukotun, K., Nayfeh, B. A., Hammond, L., Wilson, K., and Chang, K. (1996). The case for a single-chip multiprocessor. *SIGPLAN Not.*, 31(9):2–11.

Rawlins, M. and Gordon-Ross, A. (2012). An application classification guided cache tuning heuristic for multi-core architectures. In *ASP-DAC*, pages 23–28.

Renau, J., Fraguela, B., Tuck, J., Liu, W., Prvulovic, M., Ceze, L., Sarangi, S., Sack, P., Strauss, K., and Montesinos, P. (2005). Sesc simulator. http://sesc.sourceforge.net.

Staples, J. (2011). *Resource Banking: An Energy-efficient, Run-time Adaptive Processor Design Technique*. PhD thesis, University of Central Florida Orlando, Florida.

Sundararajan, K. T., Jones, T. M., and Topham, N. P. (2013). The smart cache: An energy-efficient cache architecture through dynamic adaptation. *International Journal of Parallel Programming*, 41(2):305–330.

Tarjan, D., Thoziyoor, S., and Jouppi, N. P. (2006). Cacti 4.0. Technical Report HPL-2006-86, HP Laboratories.

Weng, L. (2012). A hardware and software integrated approach for adaptive thread management in multicore multithreaded microprocessors. Master's thesis, FIU Electronic Theses and Dissertations.

Wolf, W. (2004). The future of multiprocessor systems-on-chips. In *Design Automation Conference, 2004. Proceedings. 41st*, pages 681–685.

Woo, S., Ohara, M., Torrie, E., Singh, J., and Gupta, A. (1995). The splash-2 programs: characterization and methodological considerations. In *Computer Architecture, 1995. Proceedings., 22nd Annual International Symposium on*, pages 24–36.

Youn, S., Kim, H., and Kim, J. (2007). A reusability-aware cache memory sharing technique for high-performance low-power cmps with private l2 caches. In *Low Power Electronics and Design (ISLPED), 2007 ACM/IEEE International Symposium on*, pages 56–61.