PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS
Programa de Pós-Graduação em Informática

Matheus Alcântara Souza

EXPLORAÇÃO DE ESPAÇO DE PROJETO DE ARQUITETURAS DE PROCESSADORES MANY-CORE BASEADOS EM REDES-EM-CHIP COM USO DE SIMULAÇÃO DE SISTEMAS COMPLETOS

Belo Horizonte
2015

Matheus Alcântara Souza

EXPLORAÇÃO DE ESPAÇO DE PROJETO DE ARQUITETURAS DE PROCESSADORES MANY-CORE BASEADOS EM REDES-EM-CHIP COM USO DE SIMULAÇÃO DE SISTEMAS COMPLETOS

Dissertação apresentada ao Programa de Pós-Graduação em Informática da Pontifícia Universidade Católica de Minas Gerais, como requisito parcial para obtenção do título de Mestre em Informática.

Orientador: Prof. Dr. Henrique Cota de Freitas

Belo Horizonte

2015

FICHA CATALOGRÁFICA
Elaborada pela Biblioteca da Pontifícia Universidade Católica de Minas Gerais

Souza, Matheus Alcântara
S729e Exploração de espaço de projeto de arquiteturas de processadores Many-core baseados em Redes-em-chip com uso de simulação de sistemas completos / Matheus Alcântara Souza. Belo Horizonte, 2015.
123 f. : il.

Orientador: Henrique Cota de Freitas
Dissertação (Mestrado) - Pontifícia Universidade Católica de Minas Gerais. Programa de Pós-Graduação em Informática.

1. Arquitetura de computador. 2. Simulação (Computadores digitais). 3. Multiprocessadores. 4. Redes de computadores. I. Freitas, Henrique Cota de. II. Pontifícia Universidade Católica de Minas Gerais. Programa de Pós-Graduação em Informática. III. Título.

CDU: 681.3-11

Matheus Alcântara Souza

EXPLORAÇÃO DE ESPAÇO DE PROJETO DE ARQUITETURAS DE PROCESSADORES MANY-CORE BASEADOS EM REDES-EM-CHIP COM USO DE SIMULAÇÃO DE SISTEMAS COMPLETOS

Dissertação apresentada ao Programa de Pós-Graduação em Informática da Pontifícia Universidade Católica de Minas Gerais, como requisito parcial para obtenção do título de Mestre em Informática.

Prof. Dr. Henrique Cota de Freitas – PUC Minas (Orientador)

Prof. Dr. Ricardo dos Santos Ferreira – UFV (Banca Examinadora)

Prof. Dr. Humberto Torres Marques Neto – PUC Minas (Banca Examinadora)

Belo Horizonte, 06 de julho de 2015.

A Deus, meus pais e família, e à Valéria.

AGRADECIMENTOS

Em primeiro lugar, agradeço a Deus por me guiar em todas as etapas da vida, me dando saúde e capacidade para seguir sempre em frente, ultrapassando as dificuldades. Com Ele, agradeço a toda a minha família. Aos meus pais, João e Anelize, agradeço por sempre estarem presentes ao meu lado, me mostrando o valor das coisas, e me tratando sempre com muito amor, carinho, simplicidade e razão. À minha irmã, Mariane, obrigado por compartilhar comigo bons momentos e ser a pessoa especial que é, me apoiando sempre. À Coroca, minha tia, valeu por todas as horas de descontração e por ter ajudado meus pais em nossa criação. À Valéria, obrigado por me amar e me apoiar, sempre me incentivando a seguir em frente mesmo nas horas mais difíceis. Amo vocês!

Agradecimento especial ao Prof. Dr. Henrique Cota, meu orientador, que me acolheu desde o primeiro contato no Mestrado, até o final. Sem sua sabedoria, muita inteligência estaria em falta no mundo. Obrigado pelos diálogos e discussões, e pelo tratamento igualitário e completo à todos seus alunos. Da mesma forma, agradeço aos professores do Mestrado, pelos ensinamentos durante as aulas.

À secretaria do mestrado, em especial à Giovana, obrigado por trabalharem com tanta dedicação em prol do bem estar dos alunos. Aos amigos e colegas do Mestrado, os quais não citarei nomes, mas jamais esquecerei. Agradecimento especial aos companheiros do grupo CArT (*Computer Architecture and Parallel Processing Team*) pelos momentos de estudo, discussões, diversão e trocas de experiências. Desejo sucesso à todos!

À Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES), agradeço pela bolsa concedida, e pela colaboração para o desenvolvimento do ensino no país. À Fundação de Amparo à Pesquisa do Estado de Minas Gerais (FAPEMIG) e ao *Institut National de Recherche en Informatique et Automatique* (INRIA) pelos recursos computacionais utilizados e ao Professor Jean-François Méhaut, pelo acesso disponibilizado ao MPPA-256. À Câmara Municipal de Belo Horizonte e colegas de trabalho, agradeço pela compreensão e apoio durante o Mestrado.

Por fim, agradeço a todos amigos e parentes que, de alguma forma, fizeram parte e contribuíram com mais essa jornada.

"Nós só podemos ver um pouco do futuro, mas o suficiente para perceber que há muito a fazer."

Alan Turing

RESUMO

O aumento de desempenho de processadores é um desafio que cresce a cada dia, principalmente com as preocupações relacionadas ao consumo energético dessas arquiteturas. A comunidade industrial e acadêmica tenta aplicar esforços para obter melhorias nas arquiteturas computacionais, com o intuito de remover as limitações existentes, inovando e produzindo outras alternativas. Encarar esse desafio torna-se ainda mais necessário com o crescimento da demanda por desempenho das aplicações e dos dados processados por elas.

Para atingir o desempenho computacional demandado por essas aplicações e dados, faz-se necessário buscar alternativas com alto poder de processamento, como, por exemplo, os processadores *many-core*, que fazem uso de uma rede-em-chip para integrar vários núcleos de processamento e outros componentes, como memórias. Gera-se, então, uma necessidade por métodos de estudos e experimentos dessas tecnologias. Para tanto, pode-se usar a abordagem de exploração do espaço de projeto, com o intuito de explorar várias opções de projeto de arquiteturas de computadores, organizando a pesquisa de opções. Associado à essa abordagem, pode-se usar como alternativa viável métodos de simulação de sistemas completos, possibilitando a criação de modelos de computadores virtuais contemplando todos os componentes de um sistema real, como núcleos de processamento, memórias e sistemas de interconexão, sem custos, e com alta flexibilidade.

O presente trabalho apresenta a avaliação de arquiteturas de processadores com redes-em-chip através da simulação de sistemas completos, organizada pela exploração do espaço de projeto. Os resultados mostraram que a exploração de espaço de projeto, associada com simulações de sistemas completos, pode ser utilizada, tanto pela academia quanto pela indústria, no desenvolvimento de novas opções para alto processamento.

Foram executadas cinco aplicações com diferentes padrões de comunicação e acesso à memória, em diversas arquiteturas. Essas arquiteturas foram propostas com variações na quantidade de núcleos (16, 32, 64 e 128); no tamanho de memória *cache* L2 compartilhada e distribuída (64kB, 128kB e 256kB); e, também, na topologia, sendo 12 tipos, dentre eles *Mesh*, *Torus* e *Cluster*. Ao todo, 531 simulações foram conduzidas, a partir dos requisitos e objetivos da exploração do espaço de projeto.

Constatou-se que, em alguns casos, não houve escalabilidade das cargas de trabalho quando a quantidade de núcleos foi aumentada para 64, e, em nenhum dos casos, quando aumentada para 128. Foi identificado que o tamanho de memória *cache* L2 que aparece como melhor alternativa é o de 256kB por núcleo, encontrado em 9 dos melhores casos.

Um modelo de topologia de rede-em-chip por agrupamentos (*Clusters*) foi configurado, surgindo como alternativa para processadores de alto desempenho. Com essa topologia, foi possível obter ganhos de desempenho de até $41,75\%$, em relação à uma topologia *Mesh* tradicional. Além disso, com a topologia *Cluster*, foram verificadas reduções no consumo energético de até $42,21\%$. Considerando cada aplicação executada, e quantidade de núcleos, dentre os 17 melhores resultados, a topologia *Cluster* aparece em 11 casos.

Palavras-chave: *Many-core*. Simulação. Exploração de Espaço de Projeto. Redes-em-chip.

ABSTRACT

To improve the performance of processors is a challenge that increases every day, especially as far as energy efficiency of these architectures is concerned. The industrial and academic community tries to apply efforts on their researches to obtain improvements in computer architectures, in order to take out the existing limitations, by innovating and producing new alternatives. To face this challenge becomes even more necessary with the demand for application performance and the growth in the data processing.

To achieve the computing performance demanded by these applications and data, it is necessary to use processors with high processing power, such as many-core processors, composed by a network-on-chip to integrate multiple processing cores and other components, such as memories. Thus, there is a need for tests and experiments of these technologies. Therefore, the design space exploration approach can be used, so as to explore various design options of computer architectures, organizing the options available. Associated with this approach, the full system simulation, which is a viable alternative method, may be used, enabling the modeling of virtual computers, covering all the components of a real system, such as processing cores, memories and interconnection systems, at no cost and with high flexibility.

This present paper aims to evaluate the processor architectures with network-on-chip through the simulation of full systems, organized by the design space exploration. The results showed that the design space exploration associated with the full system simulations, may be used by both academia and industry in the development of new options for high-performance computing.

Five applications, with diverse communication and memory access patterns, were executed over different architectures. These architectures were proposed varying in number of cores (16, 32, 64 and 128); in shared and distributed L2 cache memory size (64kB, 128kB and 256kB); and in 12 types of topologies as well, including Mesh, Torus and Cluster. In total, 531 simulations were conducted, considering the requirements and objectives of the design space exploration.

In some cases, the workloads did not scale well when the amount of cores was increased to 64, and, in no case, when increased to 128. It was identified that the L2 cache size seems to be the best alternative was 256kB per core, found in nine out of ten of the best cases.

A model of network-on-chip topology using clusters has been set up, which is an alternative to high-performance processors. With this topology, it was possible to obtain

performance gains up to 41.75%, compared to a traditional Mesh topology. Moreover, with the Cluster topology, up to 42.21% reduction in energy consumption was obtained. Considering each simulated application and the number of cores, the cluster topology appears in 11 cases among the 17 best results.

Keywords: *Many-core*. Simulation. Design Space Exploration. Networks-on-chip.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

BSD *Berkley Software Distribution*

CG *Conjugate Gradient*

CMP *Chip* Multiprocessado

DSE Exploração do Espaço de Projeto, do inglês *Design Space Exploration*

EP *Embarrassingly Parallel*

FLITs *Flow control digits*

FT *Fast Fourier Transform*

GCC *GNU Compiler Collection*

GPU *Graphics Processing Unit*

IS *Integer Sort*

MG *Multi-Grid*

MIMD *Multiple instructions, multiple data*

MISD *Multiple instructions, single data*

MPPA *Multi-Purpose Processor Array*

NASA *National Aeronautics and Space Administration*

NPB NAS *Parallel Benchmark*

NR *Nearest Neighbor Router*

SESC *Super Escalar Simulator*

SMP *Symmetric Multiprocessor*

SIMD *Single instruction, multiple data*

SISD *Single instruction, single data*

SUMÁRIO

1 INTRODUÇÃO

Obter o melhor desempenho possível é um dos principais objetivos durante o projeto da arquitetura de um processador. Estudos e pesquisas para obtenção de melhorias nessas arquiteturas e redução dos limites existentes têm sido realizados com o propósito de prover soluções que atendam a demanda por desempenho de aplicações desenvolvidas. O crescimento do volume de dados para processamento por meio dessas aplicações é mais um fator que demanda uma necessidade de evolução das arquiteturas de processadores, que chegaram em um patamar no qual obter mais desempenho, através do aumento da frequência dos processadores, tornou-se impraticável.

Para resolver este problema, são propostas as arquiteturas paralelas, dentre elas, as arquiteturas de sistemas distribuídos, como *Clusters* e *Grids* computacionais. Nesses modelos, vários nós computacionais são interconectados e se comunicam, aumentando a capacidade de processamento. Também foram adotadas outras abordagens, como a de *Symmetric Multiprocessor* (SMP), que consiste em ter um nó computacional com mais de um processador. Além dos SMPs, os chamados *Chips Multiprocessors* (CMPs), também conhecidos como *chips multi-core*, possuem mais de um núcleo em um único *chip*.

A evolução constante permitiu alcançar um patamar de desempenho na escala de *petaflops*, com o advento de arquiteturas paralelas mais complexas, como as *Graphics Processing Units* (GPUs), os *Field-Programmable Gate Arrays* (FPGAs), e os processadores *multi-core*. Estes tipos de arquiteturas podem compor sistemas heterogêneos, com processadores de diferentes propósitos e capacidades de processamento.

Para atingir o desempenho computacional na escala de *exaflops*, ultrapassando a barreira de *petaflops*, faz-se necessário o uso de processadores de alto desempenho com baixo consumo de energia. Estabelecida essa necessidade, esta dissertação pretende explorar as arquiteturas de processadores *many-core*. Os processadores *many-core* possuem elevada quantidade de núcleos, e.g. 128, e utilizam um sistema de comunicação intra-chip mais complexo, tal como as redes-em-chip.

Na abordagem de processadores *many-core* é necessário levar em consideração estratégias para alocar os vários núcleos e outros componentes, como memórias e barramentos, dentro de um único *chip*, aumentando a eficiência do processador (SHALF; DOSANJH; MORRISON, 2011; SIMON, 2012; ASANOVIC et al., 2009). Para a comunicação desse grande número de processadores, o uso de abordagens simples e clássicas, como barramentos, torna-se inviável, devido às restrições impostas pelos fios, ou interconexões, para comunicação no circuito eletrônico. Desta forma, uma rede de comunicação que seja eficiente pode ser aplicada, que é o caso das redes-em-chip (FREITAS; ALVES; NAVAUX, 2009). O princípio de uma rede-em-chip é reduzir o tamanho das interconexões,

adicionando roteadores intermediários para a comunicação entre os núcleos, aumentando a escalabilidade do *chip* permitindo que este contemple mais núcleos de processamento em paralelo (GUERRE et al., 2010; BJERREGAARD; MAHADEVAN, 2006). É possível ainda, com a rede-em-chip e os núcleos de processamento interconectados, propor diferentes formas de organizações desses e de outros componentes dentro do *chip*. Unidades de memória, por exemplo, podem ser distribuídas no *chip* e conectadas à rede-em-chip, na tentativa de otimizar o acesso à elas, ou até mesmo favorecer o processamento de determinada aplicação por meio do mapeamento de processos.

A escalabilidade propiciada pelo uso de redes-em-chip traz um vasto leque de opções para evolução das arquiteturas de processadores. Então, para essa evolução, há uma demanda por métodos de estudos e experimentos dessas tecnologias. Para que esse vasto leque de opções seja devidamente percorrido, pode-se usar a abordagem de Exploração do Espaço de Projeto, do inglês *Design Space Exploration* (DSE). A DSE é uma atividade de verificação e análise de um conjunto variado, e normalmente extenso, de alternativas de projeto, durante a construção, por exemplo, de um sistema computacional. Trata-se de uma abordagem usada para prototipação rápida, otimização de métricas de projeto e integração de sistemas computacionais, identificando configurações possíveis que respeitam determinadas restrições e requisitos (KANG; JACKSON; SCHULTE, 2011). Com a DSE, é possível então explorar várias opções de projeto de arquiteturas de computadores, e organizá-las, classificando as indicadas para determinado propósito.

A produção de processadores com muitos núcleos não é tarefa trivial ou de baixo custo, podendo ser inviável o estudo por meio de componentes reais, principalmente quando pretende-se reproduzir as várias opções no espaço de projeto. Como alternativa viável, propõe-se o uso de métodos de simulação, nos quais é possível criar modelos de processadores virtuais, graças a ferramentas disponíveis para a comunidade de Ciência da Computação. Entretanto, modelos de simulação tradicionais, que não observam grande parte dos elementos de um ambiente real, tendem a omitir ou distorcer resultados. Os simuladores de sistemas completos, como o *Gem5* (BINKERT et al., 2011), proporcionam uma estratégia que considera muitos fatores de um ambiente mais realista.

Os simuladores de sistemas completos permitem a simulação de um sistema eletrônico, ou de arquiteturas de computadores, em níveis de detalhes próximos dos sistemas reais. Esse tipo de simulação permite que aplicações sejam executadas sem grandes modificações, em relação à execução em um ambiente real, reduzindo o esforço de adaptação do código e alterações no comportamento da aplicação. Um simulador de sistema completo normalmente inclui como seus componentes os núcleos de processamento, memórias, redes de interconexão ou barramentos, dentre outros existentes em uma arquitetura multiprocessada. Outra vantagem desse tipo de ferramenta está na possibilidade de simular o sistema operacional, permitindo análises com suas influências.

Ainda existem muitas limitações e gargalos nas arquiteturas *many-core* que necessitam ser estudados, como o comportamento das redes de interconexão e dos componentes de memória. Há a necessidade de explorar o espaço de projeto de arquiteturas de processadores, no sentido de melhorar o desempenho e reduzir o consumo energético. Ainda que o estudo dos componentes de forma isolada seja possível, não é ideal desconsiderar a visão do todo. Desta forma, o presente trabalho propõe o uso de simulação de sistemas completos para a identificação de gargalos de processamento em arquiteturas *many-core* com redes-em-chip, e a proposta de melhores alternativas definidas por meio da exploração do espaço de projeto, considerando aspectos como topologia, quantidade de núcleos e tamanho e organização de memórias *cache* L2.

1.1 Problema

O estudo de técnicas e estratégias para a melhoria dos processadores *many-core*, seja no âmbito comercial ou acadêmico, por meio da construção real desses processadores com o uso de silício, não é tarefa trivial. O uso de ferramentas de simulação permite a avaliação e modificação de arquiteturas de computadores de maneira simplificada, bem como a análise do comportamento de aplicações quando executadas nessas arquiteturas (PATTERSON; HENNESSY, 2012). Processadores *many-core*, como o *Multi-Purpose Processor Array* (MPPA) (DINECHIN et al., 2013), são alvos de estudos para o aumento de desempenho e redução do consumo energético. Desenvolvedores e arquitetos esbarram em algumas limitações apresentadas nesse tipo de processador, como limitações de memória e gargalos de comunicação. Desta forma, o problema é sintetizado na seguinte pergunta: Considerando a possibilidade de um grande número de alternativas de projeto de arquiteturas de processadores *many-core*, quais características arquiteturais podem determinar novos rumos de projeto?

1.2 Hipóteses

Simuladores de sistemas completos podem permitir análises comparativas de diferentes arquiteturas de processadores *many-core* baseados em redes-em-chip. A simulação pode fornecer meios para a avaliação de processadores, elementos de memória e redes de interconexão, introduzindo maior flexibilidade no estudo e evolução das arquiteturas *many-core*.

Esse tipo de simulação pode permitir, ainda, análises do comportamento de aplicações paralelas em diversas arquiteturas, e auxiliar na concepção de outras. Essas aplicações paralelas provocam comunicações entre processos e uso de recursos e componentes do processador, como os elementos de memória e redes de interconexão, que

tendem a se tornar cada vez mais complexos. A abordagem de DSE pode contribuir para a organização e delimitação de um grande espaço de projeto de arquiteturas *many-core*, criando meios para a proposição de melhorias estratégicas nos recursos e componentes do processador, resultando, por exemplo, no aumento da eficiência energética.

1.3 Motivação

A alocação do maior número possível de núcleos em um único *chip* é a principal estratégia para o aumento de desempenho utilizada pela indústria de processadores. É possível encontrar, no mercado, processadores com centenas de núcleos, como no caso do MPPA-256, que possui 256 núcleos de processamento divididos em 16 agrupamentos e conectados por redes-em-chip (tudo em um único *chip*). A identificação de limites e gargalos em processadores desse tipo, ainda que em escalas menores de quantidade de núcleos ou configurações, é uma ação que merece destaque, principalmente no que diz respeito aos componentes de memória e interconexão, como redes-em-chip.

A simulação de arquiteturas *many-core*, com a abordagem de sistemas completos, permite avaliações conjuntas por parte de arquitetos de computadores e desenvolvedores de sistemas, para identificar configurações alternativas que produzam melhores resultados na relação entre o *hardware* e o *software*. Isso traz benefícios para a indústria e academia, que passa a ter oportunidades precisas para o desenvolvimento de processadores de propósito geral, ou para usos específicos.

1.4 Objetivos

1.4.1 Objetivo geral

O principal objetivo deste trabalho é a exploração do espaço de projeto por meio de um simulador de sistemas completos, para identificar limites e gargalos em arquiteturas de processadores *many-core*, detalhando abordagem de simulação e explorando alternativas para melhor eficiência energética.

1.4.2 Objetivos específicos

Pode-se estabelecer os seguintes objetivos intermediários:

a) Compreender a arquitetura do processador *many-core* MPPA-256 e suas limitações, para que esse possa ser espelhado em ambiente simulado, ainda que parcialmente.
b) Estudar, por meio da leitura e prática, o simulador de sistemas completos *Gem5* (BINKERT et al., 2011), para análise de componentes da arquitetura, como unidade

de processamento, memória e rede de interconexão.

c) Delimitar e explorar o espaço de projeto de arquiteturas de processadores *many-core*, por meio da simulação de sistemas completos, variando a quantidade de núcleos, o tamanho das *caches* de nível 2, e as topologias da rede-em-chip dessas arquiteturas.

d) Avaliar os resultados obtidos por meio de análises comparativas, definindo os rumos de projeto com foco em eficiência energética das arquiteturas, através das alternativas selecionadas.

1.5 Organização da dissertação

Esta dissertação está organizada da seguinte maneira: O Capítulo 2 apresenta uma revisão literária de temas relacionados ao trabalho proposto, bem como alguns trabalhos correlatos. No Capítulo 3, detalha-se a metodologia de pesquisa desta dissertação indicando decisões de projeto e abordagens selecionadas. O Capítulo 4 apresenta a proposta de arquiteturas para simulação. No Capítulo 5, são apresentados e avaliados os resultados dos experimentos. Por fim, o Capítulo 6 encerra este trabalho com as conclusões e trabalhos futuros.

2 REVISÃO DA LITERATURA

2.1 Arquiteturas paralelas

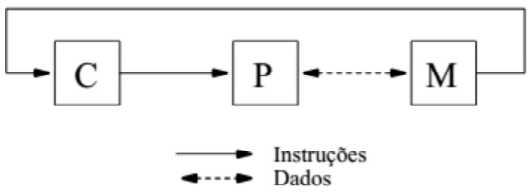
A evolução dos *chips* com um único núcleo de processamento chegou em um patamar no qual obter mais desempenho tornou-se quase impossível. Por outro lado, a demanda por alto desempenho das novas aplicações e grandes volumes de dados continua em crescimento. Há alguns anos, a indústria anunciava *chips* que contemplavam mais de um núcleo de processamento, uma vez que não era possível mais aumentar a frequência dos processadores. Esses *chips* são conhecidos como *chips multi-core* e *many-core*.

Nos últimos anos, a proposta é que milhares de núcleos de processamento sejam adicionados em um único *chip*, demandando novas arquiteturas e modelos de programação. Aumentar o número de núcleos de processamento em um único *chip* é uma alternativa que prevalece nas últimas décadas (ASANOVIC et al., 2009; HENNESSY; PATTERSON, 2007). Com esse ambiente multiprocessado, há a necessidade de programação paralela para uso pleno dos vários núcleos de processamento.

O pensamento a respeito da execução de tarefas paralelas vem desde 1966, quando Michael J. Flynn propôs uma classificação para computadores, com base no fluxo de dados e instruções durante a execução de um programa. Essa classificação é conhecida como a Taxonomia de Flynn (FLYNN, 1966), que ainda é usada para a classificação de arquiteturas paralelas. Essa taxonomia é dividida em 4 categorias:

a) *Single instruction, single data* (SISD) - Categoria em que se enquadram os *chips* monoprocessados, onde um único dado é processado por vez por uma única instrução, conforme Figura 1.

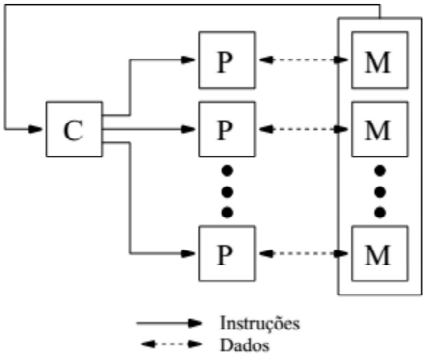
Figura 1 – Taxonomia de Flynn - SISD



Fonte: ROSE; NAVAUX, 2002

b) *Single instruction, multiple data* (SIMD) - Nesta categoria, uma única instrução é executada por várias unidades de processamento simultaneamente, com diferentes fluxos de dados, conforme Figura 2.

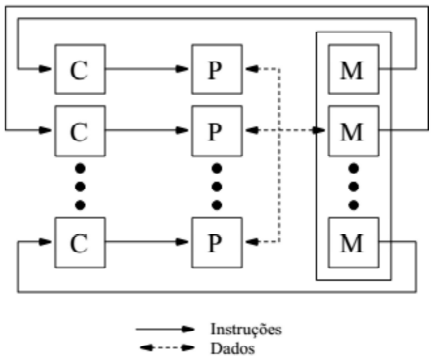
Figura 2 – Taxonomia de Flynn - SIMD



Fonte: ROSE; NAVAUX, 2002

c) *Multiple instructions, single data* (MISD) - Processadores nesta categoria, em teoria, executariam cada um sua própria instrução em um mesmo dado, conforme Figura 3. Todavia, não é um tipo de processamento comum, e não há processador comercial que seja classificado nesta categoria.

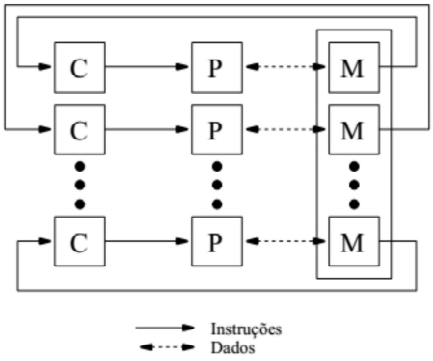
Figura 3 – Taxonomia de Flynn - MISD



Fonte: ROSE; NAVAUX, 2002

d) *Multiple instructions, multiple data* (MIMD) - Nesta categoria, cada unidade de processamento executa suas próprias instruções em um conjunto de dados diferente das outras unidades, conforme Figura 4.

Figura 4 – Taxonomia de Flynn - MIMD



Fonte: ROSE; NAVAUX, 2002

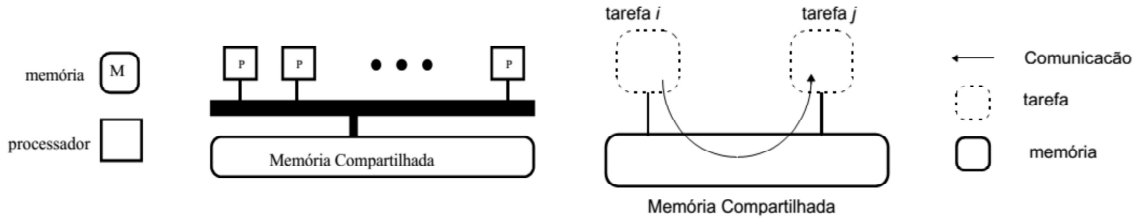
Dentre todas as categorias, a MIMD é considerada a mais flexível e vantajosa de todas, em casos gerais. Assim sendo, é a categoria usada nas arquiteturas paralelas discutidas neste trabalho.

2.1.1 Memória compartilhada e distribuída

A leitura da Taxonomia de Flynn no que diz respeito ao fluxo de dados, além do próprio advento dos *chips* multiprocessados, leva à reflexão de como esses dados são alocados para uso pelo processador. Nesse contexto estão dois paradigmas de multiprocessadores, sendo um relativo à memórias compartilhadas e outro à memórias distribuídas.

Figura 5 – Modelo de memória compartilhada e comunicação

(a) Modelo **(b) Comunicação**

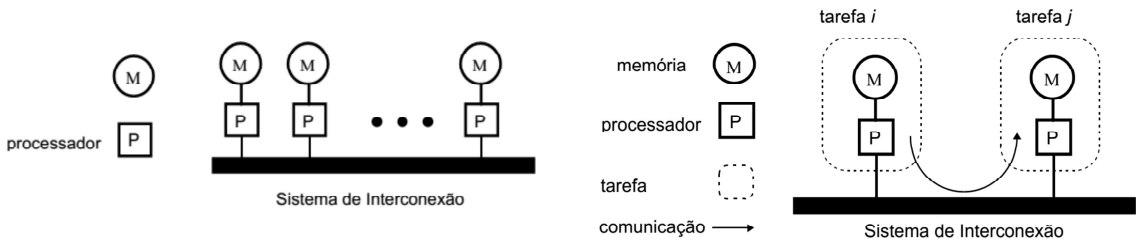


Fonte: SATO; MIDORIKAWA; SENGER, 1996

No modelo de memória compartilhada, tarefas em execução em diferentes núcleos de processamento podem acessar um mesmo endereço de memória simultaneamente. Nesse modelo existe um gargalo relativo à comunicação entre os núcleos de processamento e a memória, reduzindo a escalabilidade dessa abordagem. Em uma arquitetura com mais de 32 núcleos, por exemplo, o processamento é fortemente afetado pelo acesso concorrente à uma mesma unidade de memória. O modelo de memória compartilhada é o oposto do modelo de memória privada, onde cada núcleo de processamento somente acessa endereços de memória específicos e privados. As Figuras 5a e 5b exemplificam o modelo de memória compartilhada e a comunicação entre as tarefas, respectivamente.

Figura 6 – Modelo de memória distribuída e comunicação

(a) Modelo **(b) Comunicação**



Fonte: SATO; MIDORIKAWA; SENGER, 1996

No modelo de memória distribuída, esta é dividida entre as várias unidades de processamento, sendo o contrário do modelo de memória centralizada. Esse modelo tem maior escalabilidade que o anterior, suportando um maior número de núcleos de processamento. Para que exista a comunicação entre as tarefas em execução em cada conjunto de processador e memória, faz-se necessário o uso de uma rede de interconexão entre eles. Essa necessidade de comunicação torna o modelo de memória distribuída mais complexo para o desenvolvimento de programas. As Figuras 6a e 6b exemplificam, respectivamente, o modelo de memória distribuída e também a comunicação entre as diversas tarefas.

Uma das vantagens no modelo de memória compartilhada está no fato de que ele oferece um espaço de endereçamento de memória único para os processadores, isto é, um único lugar em que os dados podem ser encontrados. Isso facilita o desenvolvimento de aplicações paralelas, uma vez que o programador não precisa, a priori, se preocupar com a comunicação de dados entre memórias. Por outro lado, o modelo de memória distribuída possui a vantagem de oferecer escalabilidade, permitindo quantidades maiores de memória, mas a comunicação aparece como um desafio para os desenvolvedores.

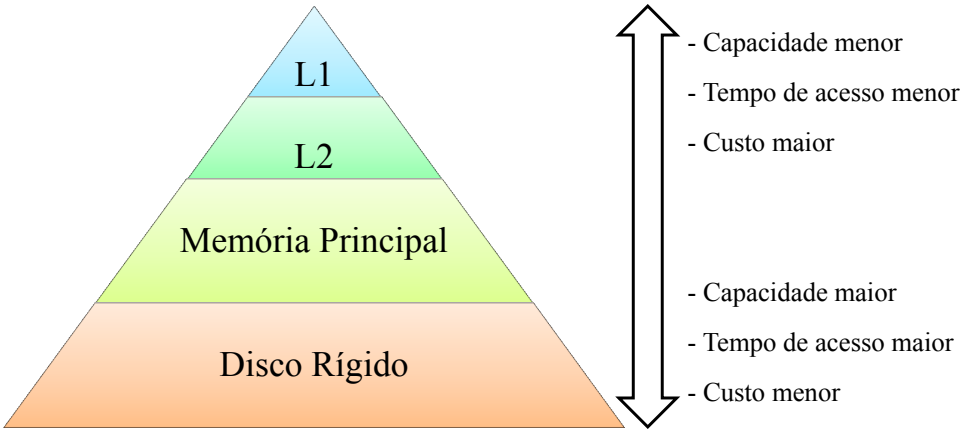
2.1.2 Memórias cache

Nos dois modelos discutidos na Seção 2.1.1, pode ser encontrada uma hierarquia de memória, que consiste basicamente de vários níveis de unidades de memória, com diferentes tamanhos e tempos de acesso. A vantagem de ter uma memória de tamanho grande, está no fato de que mais dados podem ser alocados nela. Entretanto, o aumento da memória implica em um tempo maior para identificar um dado nessa memória. Para resolver esse *trade-off*, é proposta a hierarquia de memória, com uso de *caches*.

Cache é elemento de memória localizado entre o processador e a memória principal

de uma arquitetura. Sua função é permitir que os dados sejam acessados ou escritos mais rapidamente pelo processador, mas oferecendo um espaço de memória menor. Essa redução do tamanho está atrelado ao custo da tecnologia para se fabricar um elemento desse tipo (PATTERSON; HENNESSY, 2012). Então, na hierarquia de memória, as *caches* são alocadas em níveis próximos ao do núcleo de processamento, favorecendo o princípio da localidade espacial. Descendo na hierarquia, podem existir *caches* de tamanho cada vez maior, e velocidades de acesso menores. Tem-se então os níveis de *cache*, como L1 (nível 1, e mais próximo do núcleo de processamento), e L2, que é o nível de *cache* explorado nesta dissertação. A Figura 7 apresenta um exemplo de hierarquia de memória, com dois níveis de *cache*, além da memória principal e o disco rígido.

Figura 7 – Exemplo de hierarquia de memória



Fonte: Elaborado pelo autor

Uma *cache* pode ser construída seguindo os mesmos princípios de memória discutidos na Seção 2.1.1, podendo ser distribuída e compartilhada entre vários núcleos de processamento. Esse compartilhamento pode causar alguns problemas, uma vez que um núcleo poderia alterar um dado que outro núcleo também acessa. Esse é o problema de coerência de *cache* (PATTERSON; HENNESSY, 2012). Para resolvê-lo, são propostos protocolos de coerência, com a função de sincronizar as operações em *caches* por meio de controle de estados. Um exemplo de protocolo de coerência é o MESI (IVANOV; NUNNA, 2001), que propõe os seguintes estados de determinado dado em *cache*: Modificado (*Modified*), Exclusivo (*Exclusive*), Compartilhado (*Shared*) e Inválido (*Invalid*).

2.1.3 Redes-em-chip

A inclusão de vários núcleos de processamento em um único *chip* não é uma proposta recente (ASANOVIC et al., 2009; HENNESSY; PATTERSON, 2007). É possível encontrar no mercado computadores para uso em estações de trabalho com algumas unidades de núcleos (e.g. *dual-core, quad-core, octa-core*). Em uma arquitetura desse

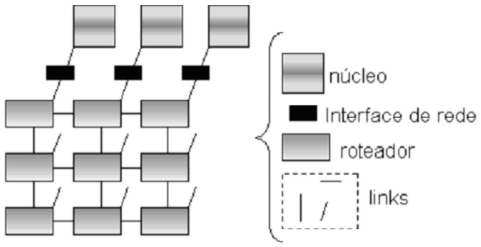
tipo é necessário existir interconexões entre os vários núcleos de processamento, tais como barramento ou chave *crossbar*. Entretanto, quando a quantidade de núcleos é maior (dezenas ou centenas), a exemplo dos *chips many-core*, esses tipos de tecnologias de interconexão tornam-se inviáveis, por limitações de escalabilidade e uso dos *links* de comunicação. Sendo assim, o uso de redes-em-chip é proposto (BENINI; MICHELI, 2002).

Uma rede-em-chip é uma rede de interconexão composta por três componentes básicos (FREITAS; ALVES; NAVAUX, 2009), listados a seguir:

a) **Roteadores:** O roteador é o principal elemento de uma rede-em-chip, uma vez que é responsável por conectar toda a rede. É função do roteador definir rotas para entrega de pacotes, controlando o fluxo e garantindo a qualidade do serviço.
b) **Interfaces de rede:** É o componente responsável por conectar cada elemento (núcleo ou periférico) a um roteador, permitindo a comunicação desses elementos com a rede.
c) ***Links* de comunicação:** São as interconexões responsáveis pela conexão dos roteadores entre si, dos roteadores com as interfaces de rede, e das interfaces com os demais elementos, formando a rede de comunicação.

A forma como os componentes da rede-em-chip são organizados, dá origem à topologia da rede. Alguns exemplos de topologias de rede são: Anel, *Mesh* e *Torus*. A Figura 8 apresenta um exemplo de rede-em-chip com topologia *mesh*.

Figura 8 – Exemplo de uma rede-em-chip com topologia *mesh*



Fonte: FREITAS; ALVES; NAVAUX, 2009

Para que aconteça o tráfego de dados na rede-em-chip, são definidas algumas políticas e estratégias de transporte, por meio de protocolos (FREITAS, 2009). Os protocolos de roteamento definem a rota que determinado pacote de dados irá seguir, na rede-em-chip. Um exemplo clássico de protocolo de roteamento é o XY, em que, dada uma topologia como a *Mesh*, o pacote percorre inicialmente as linhas, e depois as colunas, até atingir o destino.

Uma vez que os processadores *many-core* tenham se estabelecido como alternativa para o processamento de alto desempenho, as redes-em-chip podem ser apontadas como

uma ótima solução para resolver limitações relacionadas à escalabilidade e à comunicação entre núcleos de processamento. A redução do tamanho dos *links* e a presença dos roteadores resolvem a questão de escalabilidade. Entretanto, a presença de novos componentes pode trazer prejuízos em termos de consumo energético e área de ocupação do chip, que são outros desafios.

2.2 Exploração do Espaço de Projeto

Diante dos diversos tipos de configurações de arquiteturas paralelas que podem ser concebidas, graças à escalabilidade propiciada pelas redes-em-chip e às formas diversas de inserir um modelo de memória nessas arquiteturas, torna-se necessário adotar estratégias para seleção das melhores opções. Nesse aspecto, pode ser empregada a Exploração de Espaço de Projeto, ou DSE, permitindo que estudos e experimentos das arquiteturas paralelas sejam conduzidos de uma maneira organizada.

A DSE é uma atividade de verificação e análise de um conjunto variado e normalmente extenso de opções aplicáveis em um projeto. É uma abordagem que pode ser usada na definição de sistemas computacionais, para prototipação rápida, otimização de métricas de projeto e integração de sistemas (KANG; JACKSON; SCHULTE, 2011). Com a DSE é possível explorar várias alternativas de projeto de arquiteturas de computadores, e organizá-las, classificando as que respeitam determinadas restrições e requisitos sendo, assim, as mais indicadas para determinado propósito.

Em sua tese, Kunzli (2006) afirma que a DSE é um problema de otimização com objetivos antagônicos, isto é, melhorar algum aspecto normalmente irá provocar perdas em outro. O uso da DSE, então, objetiva determinar um conjunto de configurações de projeto, a partir do espaço de projeto, que possuem o melhor equilíbrio possível entre as perdas e ganhos.

A DSE é uma metodologia adequada para o estudo de arquiteturas *many-core*, auxiliando na definição dos multi-objetivos e seleção das melhores opções no espaço de projeto delimitado. Ainda no trabalho desenvolvido por Kunzli (2006), afirma-se que o uso de simulações para efetivamente executar o que é planejado na DSE produz resultados confiáveis, embora o tempo necessário para essa abordagem seja longo.

2.3 Simulação de arquiteturas de computadores

Algumas técnicas de avaliação de desempenho de sistemas computacionais podem ser utilizadas, como modelagem analítica, simulação e medição. Em Jain (1991), é apresentada uma lista de critérios a serem considerados para que se possa definir a técnica

mais adequada ao que pretende-se avaliar. Alguns desses critérios foram abordados, decidindo-se pelo uso da técnica de simulação.

Um dos critérios diz respeito ao que se espera de inovação em relação ao que já existe. A técnica de simulação permite identificar quais aspectos relacionados à arquitetura de computadores podem ser alterados e comparados ao estado atual, para identificar melhorias, em qualquer estágio do desenvolvimento de determinada arquitetura.

Outro critério está relacionado ao tempo disponível para a pesquisa. Simulações geralmente não são rápidas, porém, o tempo de pesquisa normalmente é suficiente para baterias de simulação. Além disso, o tempo pode ser otimizado com atividades de simulação paralelas.

Por fim, um último critério avaliado, se refere ao custo do uso da técnica de simulação. Existem simuladores gratuitos, e como não há produção real da arquitetura, o custo financeiro pode chegar a zero. Entretanto, faz-se necessário o uso de ambientes de computação e pesquisa adequados para a condução dos experimentos.

Dentre outras vantagens e independente da área em que se aplica, a abordagem de simulação traz a flexibilidade de controle do ambiente. É possível, com essa técnica, modelar todo um sistema que pode não existir no mundo real, favorecendo assim a inovação e descoberta de novos modelos e configurações. Mesmo que o modelo simulado exista no mundo real, pode-se usar a simulação como técnica para comparação desses modelos com outras alternativas (JAIN, 1991).

Essas possibilidades são válidas quando pretende-se simular arquiteturas de computadores. Dessa maneira, alguns simuladores para arquitetura de computadores estão disponíveis, e podem ser classificados em algumas categorias, como exemplo: simuladores de desempenho com um único processador, simuladores de desempenho com multiprocessadores, simuladores de consumo de energia e potência, simuladores modulares e simuladores de sistemas completos (YI; LILJA, 2006). Dentre todas, a categoria alvo de estudo neste trabalho é a de simuladores de sistemas completos.

2.3.1 Simuladores de sistemas completos

O desempenho geral de uma determinada arquitetura, observada a execução de cargas de trabalho científicas, é em geral concentrado no uso de núcleos de processamento e memória. Entretanto, a análise com a presença de um sistema operacional e seus subprogramas, da interconexão entre os diversos componentes da arquitetura, e de periféricos, também tem a sua importância. Este aspecto é evidenciado principalmente quando a análise é voltada para aplicações comerciais (por exemplo bancos de dados e servidores web), em que o desempenho do sistema operacional e dos componentes de

entrada e saída podem ser determinantes.

Tendo isso em mente, propõe-se a simulação de um sistema completo. Um simulador de sistemas completos é uma ferramenta que permite simular uma arquitetura de computador com certo nível de detalhes, possibilitando a execução de uma aplicação nesse ambiente sem grandes alterações em relação a execução em um sistema real, o que também se aplica a um sistema operacional. No que se refere aos componentes, um simulador de sistemas completos permite a configuração de núcleos de processamento, memórias, redes ou barramentos para interconexão e periféricos, permitindo a avaliação de influências de um componente no comportamento de outros existentes.

Dois simuladores de sistemas completos, dentre outros, podem ser citados: *Gem5* (BINKERT et al., 2011) e *Simics* (MAGNUSSON et al., 2002). O primeiro possui o tipo de licença *Berkley Software Distribution* (BSD), de código aberto, enquanto o segundo possui licença proprietária para aquisição comercial.

Outra característica de um simulador de sistemas completos é a possibilidade de integração com outros simuladores com propósitos específicos. Como exemplo, o simulador *Gem5* permite a integração com os simuladores *GARNET* (AGARWAL et al., 2009) e *ORION* (KAHNG et al., 2009). O *GARNET* permite a simulação de redes de interconexão (como as redes-em-chip), modelando, por exemplo, diversos tipos de roteadores. O *ORION* permite realizar estimativas relacionadas a área ocupada e energia consumida por redes-em-chip.

A complexidade dos simuladores de sistemas completos podem, em alguns casos, trazer dificuldades para o uso ou determinadas atividades. Como exemplo, podem existir limitações na quantidade de núcleos simulados. De toda forma, esse tipo de simulação é uma alternativa viável e suficientemente flexível para a condução deste trabalho.

2.4 Trabalhos correlatos

Trabalhos que tratam da análise de processadores com redes-em-chip são encontrados na literatura. Os casos em que a análise utiliza alguma metodologia de simulação são também numerosos. Todavia, ainda não é muito comum a existência de pesquisa em processadores *many-core* com simulação de sistemas completos que façam uma análise ampla de todos os componentes simulados nesta dissertação. Os estudos geralmente refletem um esforço de pesquisa para melhorias no roteador, o principal componente da rede-em-chip. Outras pesquisas procuram investigar as topologias de rede, identificando limitações. Os principais aspectos avaliados na maioria dos trabalhos dizem respeito ao aumento de desempenho e à redução do consumo energético.

A seguir são apresentadas publicações relativamente recentes (posteriores a 2010)

que utilizaram como metodologia de avaliação de processadores com redes-em-chip a simulação de sistemas completos com aplicação de alguma carga de trabalho.

O conceito de virtualização em redes-em-chip é discutido por Triviño et al. (2011). Neste trabalho, os autores preocupam-se com a execução simultânea de aplicações em CMPs, o que pode provocar comportamentos imprevisíveis no tráfego da rede-em-chip. A proposta apresentada isola o tráfego de cada aplicação na rede, por meio de canais virtuais. O tráfego na rede, desempenho e consumo energético foram avaliados na simulação, com execuções simultâneas de cargas de trabalho variadas.

Uma topologia de rede-em-chip diferenciada, *NR-Mesh*, é proposta por Camacho et al. (2011). Nessa topologia, um nó (ou núcleo de processamento) pode enviar uma mensagem pela rede, bem como recebê-la, através de diferentes roteadores vizinhos, ou *Nearest Neighbor Router* (NR). A proposta, que envolve novos algoritmos de roteamento, reduz a quantidade de saltos e consequentemente o tempo de tráfego das mensagens pela rede. Os resultados apontaram melhoras tanto no desempenho das cargas de trabalho, quanto na redução de consumo energético, em relação à topologia *Mesh* tradicional.

Ainda no que se refere a topologia de redes-em-chip, Xu et al. (2013) propõem uma rede-em-chip tridimensional com o objetivo de alcançar melhores índices de desempenho e consumo energético, bem como reduzir o custo de fabricação. É proposta a disposição de conexões verticais na rede-em-chip em um formato apontado como ótimo pelos autores, o que implicou em ajustes nos roteadores e *links*, bem como em novos algoritmos de roteamento e tráfego. Além da simulação tradicional para métricas de desempenho e energia, foram coletados resultados relacionados com a área de ocupação dos componentes dentro do *chip*, que apontaram as reduções nos custos de fabricação.

No trabalho desenvolvido por Udipi, Muralimanohar e Balasubramonian (2010), os autores avaliam outra topologia de rede hierárquica, em que vários núcleos de processamento são divididos em grupos. O primeiro nível da rede conecta os grupos enquanto um segundo nível conecta núcleos dentro de um grupo. O trabalho tem o objetivo de encontrar uma rede considerada ótima, no que se refere ao nível intra-grupo. Os autores então propõem e avaliam, através de simulação, o uso de barramentos como meio de interconexão intra-grupo, de maneira a anular o consumo energético dos roteadores, que seriam utilizados em outra abordagem.

Matsutani et al. (2010) propõem um controlador de componentes do roteador, em termos de consumo energético. Com esse dispositivo, os componentes do roteador são mantidos desativados e somente são acionados quando há necessidade, economizando energia. Em contrapartida, houve redução de desempenho atribuída à latência para acionar componentes em espera, bem como o aumento da área ocupada no *chip*. De maneira similar, o trabalho desenvolvido em Muralidhara e Kandemir (2011) propõe um mecanismo de controle de fluxo na rede, desativando os *links* de comunicação e acionando-os quando necessário. O controle ocorre de maneira dinâmica, o que provoca degradação de performance quando é necessário ativar algum *link*. Todavia, ganhos relacionados a consumo energético foram obtidos.

Um protocolo de roteamento para difusão seletiva e ponto-a-ponto não determinístico é apresentado por Daneshtalab et al. (2011). A proposta é que a escolha do tipo de difusão seja feita de maneira dinâmica, sem usar canais virtuais. Os resultados obtidos com a simulação de cargas sintéticas e reais apontaram redução no consumo energético em geral, bem como redução na latência inerente ao tráfego de pacotes.

Uma implementação para controle de fluxo na rede é apresentada por Chen, Wang e Pinkston (2012). Os autores observam que mecanismos de controle de fluxo que consideram características globais, como comportamento dos roteadores em nós vizinhos, podem alcançar melhores resultados do que aqueles que levam em consideração apenas características locais. A implementação proposta, quando simulada, apresentou redução na latência dos pacotes de dados e aumento de desempenho.

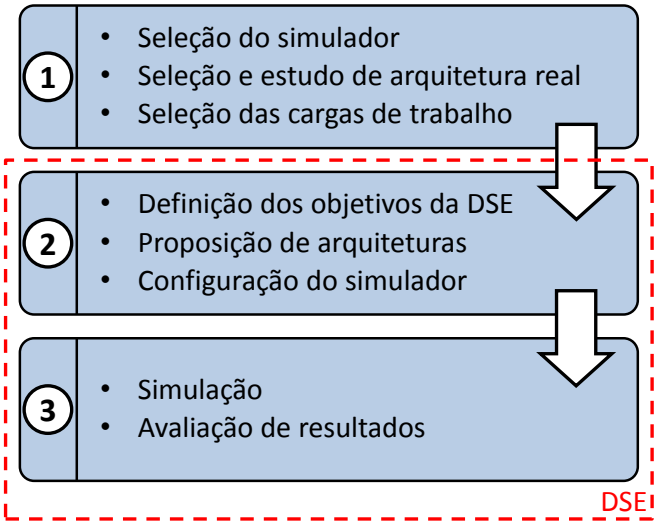
Alguns trabalhos que fazem uso da DSE em sistemas embarcados ou redes-em-chip podem ser encontrados. Como exemplo, Li e Hammami (2006) propõem uma metodologia de exploração do espaço de projeto para redes-em-chip. E um *framework* híbrido e otimizado para a exploração do espaço de projeto é apresentado em (KUNZLI, 2006).

Os trabalhos citados não esgotam o panorama atual de pesquisa em processadores com redes-em-chip, mas trazem uma boa referência de componentes estudados, ferramentas utilizadas e tendências de pesquisa, que também reforçam a importância deste trabalho. Não foi constatado, entretanto, experimentos que usam a DSE como base para definição de multi-objetivos, considerando os componentes em estudo nesta dissertação. Os dois últimos trabalhos preocupam-se mais com a forma de aplicação da DSE, do que com a exploração propriamente dita, definindo melhores opções. Não foram encontrados trabalhos que utilizem DSE para avaliar arquiteturas *many-core* com redes-em-chip, através da simulação de sistemas completos, considerando aspectos como topologia, quantidade de núcleos e tamanho e organização de memórias *cache* L2, de uma única vez, o que reforça a contribuição desse estudo nesta dissertação.

3 METODOLOGIA

Neste capítulo é apresentada a metodologia de avaliação de arquiteturas de processadores com redes-em-chip através da simulação de sistemas completos. Essa metodologia foi elaborada com base nos estudos realizados previamente, associados ao estado da arte de que trata o Capítulo 2, e está dividida em 3 macro etapas, conforme ilustrado na Figura 9. Na primeira etapa, realizou-se a escolha das ferramentas de simulação, da arquitetura a investigar e das cargas de trabalho para coleta de resultados. Em seguida, na segunda etapa, estabelece-se as configurações do simulador, produzindo assim as diferentes propostas de arquiteturas. Já na etapa três, realiza-se a simulação, definindo as métricas para a coleta e avaliação de resultados.

Figura 9 – Etapas da metodologia



Fonte: Elaborado pelo autor

A arquitetura para pesquisa e referência foi selecionada, sendo esta a do processador *many-core* MPPA-256 (DINECHIN et al., 2013), uma vez que esse processador apresenta as características discutidas neste projeto, mesclando o uso de redes-em-chip e memória compartilhada e distribuída. Em seguida, cargas de trabalho com propósitos variados foram aplicadas no estudo, para tornar possível a compreensão, durante a simulação de sistemas completos, dos comportamentos produzidos.

De posse do simulador, da arquitetura alvo e das cargas de trabalho, definiu-se o escopo da DSE, estabelecendo-se os objetivos e parâmetros a avaliar. Preparou-se, então, o ambiente para experimentação. Na sequência, a condução dos experimentos, além da coleta e avaliação dos resultados.

As próximas seções descrevem os detalhes de cada etapa, em que são justificadas as escolhas e decisões do trabalho, bem como são detalhadas as propostas para análise.

3.1 Seleção do simulador

Em um primeiro momento definiu-se as ferramentas a serem utilizadas. Ou seja, o simulador de sistemas completos, bem como simuladores e ferramentas auxiliares, para uma detalhada averiguação de potencialidades e limitações. Esta etapa permitiu também a aquisição de conhecimento prático a respeito das técnicas de simulação. Para a escolha desse simulador, três critérios principais foram avaliados, sendo: (i) tipo de licença, o que impacta diretamente na disponibilidade do simulador; (ii) a documentação disponível, que quanto mais completa, melhor para o entendimento da ferramenta; e por fim (iii) o potencial do simulador para permitir uma simulação de sistemas completos, modelando-se arquiteturas *many-core*. A pesquisa pelos simuladores disponíveis foi feita através da leitura de trabalhos que usaram tais ferramentas.

O primeiro simulador identificado foi o *Simics* (MAGNUSSON et al., 2002), que, quando associado ao simulador *GEMS* (MARTIN et al., 2005), permite a simulação de sistemas multiprocessados com um nível de detalhes muito bom. Entretanto, a licença para uso acadêmico deixou de ser válida, o que impediu a continuidade do estudo. A documentação disponível para o *Simics* foi considerada escassa, em relação aos demais. Outro simulador avaliado foi o *Super Escalar Simulator* (SESC) (RENAU et al., 2005). Nele, é possível modelar tanto processadores *single-core* quanto *chips* multiprocessados, com *pipeline* superescalar fora de ordem com predição de desvio, memórias *cache* e barramentos. O *SESC* também possui documentação limitada, mas não tem uso restrito por licença, o que permitiu o uso prático desse simulador, no trabalho desenvolvido em Souza et al. (2014). Porém, o simulador está limitado nas possibilidades de modelagem, não sendo possível, por exemplo, a configuração de uma rede-em-chip.

O simulador escolhido, dentre outras opções, foi o *Gem5* (BINKERT et al., 2011) que é uma opção viável para o desenvolvimento deste trabalho, atendendo aos critérios definidos. É um simulador com a licença do tipo BSD, com uma documentação ampla em relação aos demais. A quantidade de documentação disponível, ainda que incompleta, e a existência de uma comunidade de discussão *on-line* com boa atividade, possibilitou identificar o potencial do simulador para o trabalho proposto, e consequentemente a sua seleção como a ferramenta.

O *Gem5* é uma plataforma modular para pesquisa em arquitetura de computadores que adota um modelo de simulação determinístico, desenvolvido a partir da junção dos simuladores *M5* (BINKERT et al., 2006) e *GEMS* (MARTIN et al., 2005) por parte de instituições acadêmicas e industriais. O simulador está escrito em linguagens Python e C++, possuindo código aberto.

Nesse simulador, é possível configurar vários tipos de arquiteturas (*single-core*, *multi-core*, *multi-threading* e *many-core*) e conjuntos de instruções (ALPHA, ARM, MIPS, Power, SPARC e x86). Há, ainda, a possibilidade de simular a execução de sistemas operacionais (Linux, FreeBSD e Solaris) e aplicações diversas. Dentre os componentes que se pode parametrizar, estão a unidade de processamento e as unidades de memória (hierarquia de *caches*, memória principal e discos). Torna-se possível, então, simular configurações com diferentes números de núcleos, hierarquias de memória variadas e quantidade e tamanhos de *caches*.

A sua característica modular permite a configuração de componentes individuais e em seguida a união desses componentes através de interconexões. Essa característica torna o simulador uma ferramenta flexível, permitindo que novos componentes sejam inseridos e conectados em barramentos também configuráveis. O esquema modular do simulador, também chamado na documentação de 'sistema de memória', está dividido em dois tipos: (i) *Classic Memory System*, originado do *M5*; e (ii) *Ruby Memory System*, originado do *GEMS*. O primeiro é um esquema mais simples e de fácil configuração (feita estritamente por arquivos em linguagem Python), além de permitir tempos de simulação menores. Entretanto, o sistema clássico é menos preciso, uma vez que não permite, por exemplo, a configuração de protocolos de coerência de *cache* detalhados. O segundo esquema, *Ruby*, tem características inversas ao outro, isto é, apresenta tempos de simulação e complexidade de configuração maiores, justamente por ser mais preciso e detalhado. É importante não confundir o nome do esquema com a linguagem de programação de mesmo nome.

Para o desenvolvimento deste trabalho, optou-se pelo uso do conjunto de instruções x86; sistema operacional Linux com *Kernel* na versão 2.6.28.4; e o sistema de memória *Ruby*. A escolha do sistema de memória se justifica pela necessidade de simular uma configuração de rede-em-chip, que não é possível no outro esquema. Ainda assim, faz-se necessário o uso do *GARNET* (AGARWAL et al., 2009), um modelo de redes de interconexão detalhado, que implementa roteadores com *pipeline* de cinco estágios e um fluxo de controle por canais virtuais. Esse modelo permite a configuração de tamanho de *buffers* em nível de *Flow control digits* (FLITs), que é uma unidade de tamanho de pacotes de rede.

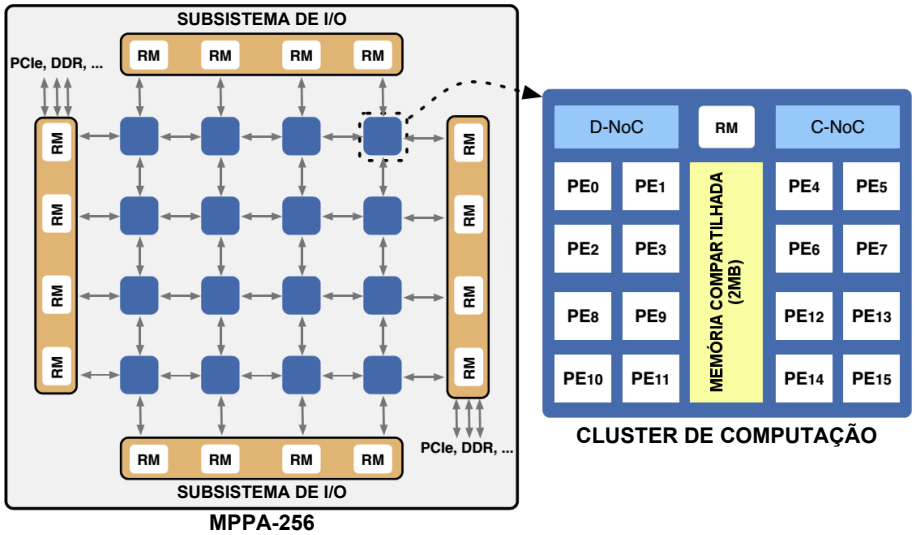
De forma complementar, usou-se o simulador *ORION* 2.0 (KAHNG et al., 2009). Este simulador é uma ferramenta para obtenção de informações sobre consumo de potência das redes-em-chip simuladas. O *ORION* tem integração com o Gem5 e o *GARNET*, de forma que os resultados de potência consumida são gerados durante a simulação.

3.2 Seleção da arquitetura alvo

No processo de seleção da arquitetura alvo, procurou-se encontrar uma arquitetura de processador *many-core*, com rede-em-chip, e elementos de memória compartilhada e distribuída. Além disso, tentou-se identificar o que a arquitetura apresenta em termos de vantagens, como por exemplo economia energética e capacidade de processamento elevada, e também suas desvantagens (limitações e gargalos dessa arquitetura). Qualquer arquitetura poderia ser escolhida como referência, entretanto, poucas possuem as características desejadas e definidas nessa dissertação.

O processador MPPA-256 (DINECHIN et al., 2013) foi escolhido nessa etapa, uma vez que é um processador que compreende um grande número de núcleos, implementa uma rede-em-chip, e possui memória distribuída e compartilhada em sua arquitetura, atingindo assim os primeiros critérios. Trata-se de um processador *many-core* desenvolvido pela empresa Kalray, que engloba 256 núcleos de processamento em um único chip. Uma visão geral desse processador é apresentada na Figura 10. O processador implementa uma organização em grupos de computação (ou *clusters*) totalizando 16 grupos. Cada *cluster* contempla 16 núcleos de processamento (representados por PE_0–PE_{15}) e um gerenciador de recursos (RM). Esses *clusters* possuem ainda, cada, uma unidade de memória de 2 MB, compartilhada entre os 16 núcleos do *cluster*.

Figura 10 – Visão geral do MPPA-256



Fonte: CASTRO et al., 2013

Os núcleos de processamento, cada um com uma frequência de 400 MHz, são responsáveis pela execução das *threads* criadas pelo usuário ou programa em execução, sendo uma *thread* por núcleo. Cada núcleo possui uma *cache* L1 de 8 kB para dados, e 8 kB para instruções, ambas com associatividade por conjunto de 2 vias. Para comunicação com a memória principal e outros dispositivos, como por exemplo interfaces

de rede e PCI, são implementados 4 outros grupos, denominados subsistemas de I/O. Um grupo de computação não acessa a memória principal ou a memória de um outro grupo diretamente. Sendo assim, todos os grupos e os subsistemas de I/O são conectados por duas redes-em-chip paralelas, com links bi-direcionais, sendo uma para dados (D–NoC) e outra para controle (C–NoC). A rede-em-chip é implementada com a topologia *Torus*.

O compilador utilizado para a arquitetura do MPPA-256 é específico da arquitetura. Esse compilador implementa uma interface para que o programador defina as comunicações entre os *clusters* de computação, e suporta bibliotecas para programação paralela como OpenMP e POSIX Threads. Não foi necessário usar o compilador para essa arquitetura, já que os experimentos foram realizados por meio de simulações.

Alguns trabalhos que fizeram uso do MPPA-256 o consideram um processador próprio para computação de alto desempenho e com baixo consumo de energia. Por outro lado, esses mesmos trabalhos apontam algumas das dificuldades para o desenvolvimento de aplicações e limitações da arquitetura (CASTRO et al., 2013; FRANCESQUINI; GOLDMAN; MÉHAUT, 2013; FRANCESQUINI et al., 2015).

Essa arquitetura apresenta um sistema de memória limitado em 2 MB por *cluster*. Cada unidade de memória, compartilhada por 16 *threads*, pode ser considerada pequena. Com essa restrição, é necessário que um alto número de comunicações seja feito através da rede-em-chip, produzindo um gargalo de comunicação entre os 16 grupos e os 4 subsistemas de I/O. Dessa maneira, pretende-se explorar essas características através da simulação de sistemas completos, avaliando as vantagens e desvantagens de arquiteturas similares no que se refere à topologia e sistema de memória. Nessa exploração, algumas características do processador MPPA-256 não são reproduzidas, como a quantidade de núcleos ou acesso à memória principal, conforme explicações na Seção 3.4.3.

3.3 Seleção das cargas de trabalho

De maneira a permitir a avaliação de arquiteturas e do comportamento de aplicações de propósito geral, algumas cargas de trabalho foram selecionadas. O principal critério para seleção das cargas baseia-se na diversidade de padrões de comunicação e paralelismo de dados. Outra característica está na necessidade de que as cargas de trabalho sejam implementadas com paralelismo (*multi-thread*) para explorar os diversos núcleos de processamento da arquitetura *many-core*.

Optou-se pelo uso do NAS *Parallel Benchmark* (NPB) (BAILEY et al., 1991), um conjunto de programas desenvolvidos pela *Advanced Supercomputing Division* da *National Aeronautics and Space Administration* (NASA), para auxiliar na avaliação de supercomputadores com arquiteturas paralelas. Foram selecionadas cinco aplicações do

NPB, nomeadas pelas siglas *Conjugate Gradient* (CG), *Embarrassingly Parallel* (EP), *Fast Fourier Transform* (FT), *Integer Sort* (IS) e *Multi-Grid* (MG). Estas aplicações compreendem um bom conjunto de características, como detalhado a seguir:

- CG (*Conjugate Gradient*): Utiliza um método de gradiente conjugado para encontrar uma estimativa do menor autovalor de uma matriz simétrica positiva, usando o método do gradiente conjugado. Caracterizada por comunicações e acessos à memória irregulares.
- EP (*Embarassingly Parallel*): Aplicação que produz um conjunto de pares gaussianos aleatórios e independentes, organiza-os e produz como resultado a combinação das somas das várias unidades paralelas, usando o método polar de *Marsaglia*. Esta aplicação não possui comunicações ou acessos a memórias em padrões significantes.
- FT (*Fast Fourier Transform*): Essa aplicação propõe-se a resolver numericamente equações parciais diferenciais utilizando transformadas tridimensionais de *Fourier*. Trata-se de uma aplicação que produz comunicações todos-para-todos, em grande volume.
- IS (*Integer Sort*): É uma aplicação robusta que realiza operações de ordenação de números inteiros, usando o método *bucket sort*. Uma característica desta aplicação está na forma de acesso à memória, que são acessos feitos de maneira aleatória.
- MG (*Multigrid*): Por essa aplicação é possível obter uma solução aproximada para uma equação tridimensional discreta de *Poisson*, utilizando o método multigrade Ciclo-V. É caracterizada pelas comunicações de longa e pequena distância, e também é considerada uma aplicação que produz acessos intensos à memória.

A versão selecionada do NPB foi a 3.3, classe *S*, com uso da biblioteca OpenMP. As aplicações relacionadas foram compiladas para uso no simulador *Gem5*. Utilizou-se o compilador *GNU Compiler Collection* (GCC), em sua versão 4.2.1, com compatibilidade para o *Kernel Linux* na versão 2.6.28.4.

3.4 Escopo da DSE

A segunda etapa da metodologia compreende a concepção das opções de arquiteturas que endereçam o problema especificado. Para tanto, tem-se como requisitos das arquiteturas as características selecionadas do processador MPPA-256, conforme descrito na Seção 3.2, que são: (i) memórias compartilhadas e distribuídas; (ii) redes-em-chip; e (iii) *many-core*. Além dos requisitos, tem-se as ferramentas, que

compreendem o simulador *Gem5* e as cargas de trabalho do *NAS Parallel Benchmarks*, descritos nas seções 3.1 e 3.3, respectivamente. A Figura 11 apresenta a visão geral da DSE aplicada nesta dissertação.

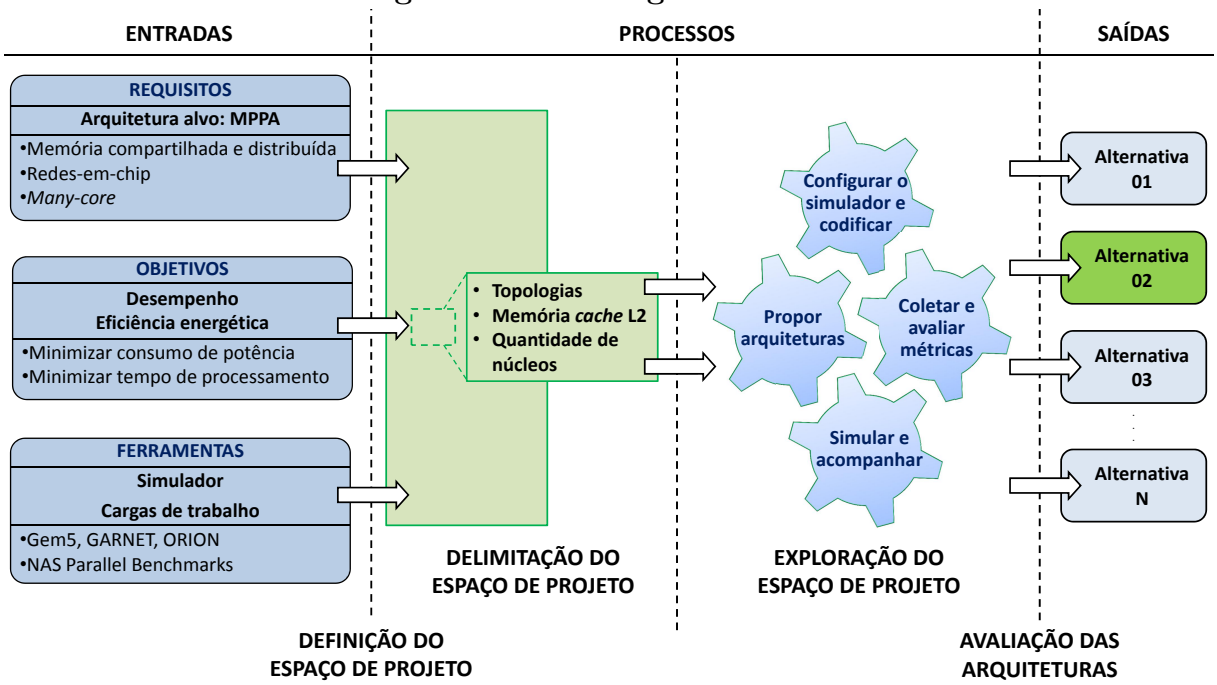
Outra entrada necessária, como pode ser notado na primeira parte da Figura 11, corresponde aos objetivos da DSE, que serão definidos na Seção 3.4.1, a seguir.

3.4.1 Objetivos da DSE

No contexto da exploração de espaço de projeto, são listados alguns objetivos a seguir, que compreendem minimizar algumas métricas:

a) Minimizar o tempo de processamento da aplicação (t): É o tempo que a aplicação leva para ser processada na arquitetura simulada.
b) Minimizar a potência consumida (p): Quantidade de potência consumida no uso da rede-em-chip, durante o processamento da aplicação na arquitetura simulada.

Figura 11 – Visão geral da DSE



Fonte: Elaborado pelo autor

Minimizar o tempo de processamento da aplicação é um objetivo que está alinhado com os desafios por maior desempenho e performance dos processadores, como discutido no Capítulo 1. As alterações para alcançar melhorias de desempenho normalmente aumentam a complexidade e robustez das arquiteturas, que passam a consumir mais energia. Minimizar a potência consumida é um objetivo que também está em acordo com os desafios estabelecidos.

A meta maior da DSE aqui desenvolvida é encontrar a arquitetura cujo produto $e = t \times p$ seja o menor possível, ou seja, reduzir o consumo energético, dado por e. Pretende-se coletar do simulador *Gem5* as métricas t e p, para, então, calcular a métrica e, que não é produzida diretamente pelo simulador. Com essas três métricas (tempo, potência e energia), será possível verificar, dentre um conjunto de alternativas de arquiteturas definidas na DSE, quais oferecem maior desempenho e maior eficiência energética, dada uma carga de trabalho.

E com o intuito de auxiliar na análise das métricas descritas, outras métricas secundárias, que não são tratadas como objetivos da DSE, serão coletadas também do *Gem5*:

a) Latência de rede média: Ciclos de atraso na entrega dos dados pela rede.
b) Taxa de faltas em *cache*: Percentual de vezes que um dado não foi encontrado em determinada *cache*, durante um acesso.

Observando o requisito 'redes-em-chip', espera-se que durante a DSE, alternativas variadas dessas redes de interconexão sejam exploradas. Cada alternativa de rede-em-chip, quando alterada, por exemplo, a topologia, terá um impacto diferente na performance e consumo energético do processador (ZEFERINO, 2003). A latência de rede é uma métrica que irá demonstrar qual o impacto da rede-em-chip no desempenho final do processador, uma vez que o atraso na comunicação dos dados também atrasa o processamento geral.

Com referência às memórias, especificamente as memórias *cache*, a taxa de faltas (*cache miss*) é uma métrica relevante. Com ela, é possível identificar quantos acessos à *cache* foram atrasados, devido à inexistência de um dado nesta cache. Essa falta de dados resulta em acessos aos níveis inferiores de memória mais lentos, causando, consequentemente, o aumento do tempo de processamento. Cada acesso aos níveis inferiores significa que mais componentes e redes de interconexão do processador serão usados, causando também o aumento do consumo de potência e energia. Sendo assim, a métrica de taxa de faltas irá auxiliar no entendimento da influência de cada alternativa de memória *cache* explorada na DSE na performance e eficiência energética do processador.

As métricas foram coletadas, calculadas e avaliadas na etapa de exploração do espaço do projeto ilustrada na Figura 11.

3.4.2 Configuração das arquiteturas

Estabelecidas as entradas da DSE, tem-se a definição do espaço de projeto. Esse espaço pode tender ao infinito, uma vez que inúmeras alternativas de projeto podem ser concebidas, se não existir uma delimitação de escopo. Na Figura 11, o espaço de projeto é representado por um universo maior, no qual é feita uma delimitação, definindo-se o escopo

do processo. Essa delimitação compreende a definição de quais itens serão verificados, e de que forma, para propor alternativas arquiteturais viáveis para simulação.

Inicialmente, propõe-se a alteração do número de núcleos. Com essa variação, será possível entender se o aumento de núcleos provocará melhora no desempenho, o que é esperado, já que a quantidade de recursos para processamento disponível será maior. Além disso, será possível validar a escalabilidade facilitada pelo uso das redes-em-chip. Ainda assim, o aumento no número de núcleos pode provocar um aumento na comunicação, gerando gargalos na rede-em-chip, ou em memória, dado o acesso concorrente de vários núcleos a um mesmo recurso.

Outro ponto de avaliação são os componentes de memória. Como observado durante o estudo do processador MPPA-256, foi identificado um problema relativo ao tamanho da memória de 2 MB, totalizando 32 MB nesse processador. Para tentar aproximar essa configuração, propõe-se a análise com memórias *cache* de segundo nível, ou L2. Essa avaliação torna-se importante a partir do momento em que apenas aumentar o tamanho da memória pode não ser uma boa opção. Os trabalhos que referenciam o MPPA-256 questionam o tamanho reduzido, mas, como foi verificado no trabalho desenvolvido em (SOUZA et al., 2014), aumentar o tamanho da *cache* não é sempre a melhor alternativa, podendo piorar o desempenho, dado o aumento da latência de acesso à *cache*, ou características intrínsecas das aplicações. Por outro lado, a quantidade de faltas em *cache* pode ser reduzida, graças ao espaço maior disponível, melhorando o desempenho.

No que se refere à topologia, que é do tipo *Torus* no MPPA-256, pretende-se explorar, inicialmente, 3 tipos. O primeiro deles, a topologia *Mesh*. Também será avaliada a própria topologia Torus, semelhante à usada no MPPA-256 para interconexão dos *clusters*. É importante ressaltar que, no MPPA-256, a rede-em-chip é implementada para os *clusters*, não estando diretamente conectada pelos roteadores aos núcleos de processamento. Dessa maneira, optou-se por avaliar também a topologia do tipo *Cluster* como alternativa para simulação. Esse tipo de topologia pode reduzir a influência da rede-em-chip durante a execução de aplicações, uma vez que ela reduz os saltos entre núcleos que estariam distantes em uma topologia tradicional, como a *Mesh* (FREITAS, 2009). Dessa forma, possíveis ganhos em termos de performance podem ser obtidos, assim como reduzir o consumo de potência.

Outra alternativa é variar cada uma das topologias, alterando por exemplo o posicionamento da *cache* L2 e dos núcleos. A partir de uma topologia *Mesh* tradicional, na qual cada roteador conecta-se a um núcleo e a uma unidade de *cache* L2, pode-se alterar o tamanho das *caches* e diminuir a quantidade existente. Com essa proposta, cada núcleo seria conectado a um roteador não compartilhado com outro componente. As *caches* seguem a mesma ideia, sendo conectadas a roteadores que não estão conectados

a núcleos. Essa estratégia, em teoria, pode aumentar o tráfego na rede, visto que para um núcleo acessar uma *cache*, pelo menos dois roteadores precisariam ser acionados. Por outro lado, o aumento do tamanho pode favorecer o compartilhamento de dados.

3.4.3 Avaliação das arquiteturas

O espaço de projeto delimitado nesta dissertação será explorado através de simulações de sistemas completos. Esse processo é composto de outros subprocessos ilustrados na Figura 11. Um deles é a proposta das alternativas de arquiteturas, que será detalhada no Capítulo 4.

Outro subprocesso envolve transicionar o que está projetado para o ambiente de simulação do *Gem5*. Essa etapa pode ser feita de duas formas básicas: a primeira compreende a definição de parâmetros do *Gem5*, com os valores desejados; a segunda compreende a escrita de códigos, a fim de incluir na arquitetura componentes específicos para simulação. A execução completa do simulador, quando não houver erro, acontecerá apenas uma vez para cada configuração. Isto pelo fato de que o *Gem5* é um simulador determinístico, portanto, simular uma configuração idêntica, novamente, produz os mesmos resultados. Com isso, os resultados apresentados não tem informações de desvio padrão.

Durante a inspeção e configuração do simulador, algumas restrições foram encontradas. Em um primeiro caso, constatou-se que não é possível simular configurações com mais do que 128 núcleos, como no processador MPPA-256, por limitações do *kernel* do sistema operacional utilizado pelo *Gem5*. Para que isso fosse possível, seria necessário compilar um novo *kernel*, alterando parâmetros e configurações. Esta tarefa pode ser realizada em trabalhos futuros.

Outra restrição está no fato de que a arquitetura do processador MPPA-256 e suas limitações, detalhadas na Seção 3.2, são bem específicas, dificultando a réplica de todos os detalhes no simulador. Como exemplo, a comunicação direta com a memória fora do *chip*, que não existe no MPPA-256, e irá ocorrer no simulador. Para reproduzir esse comportamento, seria necessário alterar profundamente o código do simulador, o que também pode ser feito em trabalhos futuros. De toda forma, o estudo não será prejudicado, já que a modularidade do *Gem5* permite a análise dos componentes de maneira isolada.

Com relação à alteração do tamanho das memórias *cache*, foi discutido na Seção 3.4.1 a respeito da lentidão provocada pelo acesso à memórias de nível inferior pela ocorrência de um *cache miss*. Essa lentidão refere-se à latência de acesso, que é maior quando os tamanhos de memória são maiores (PATTERSON; HENNESSY, 2012). Dessa forma, quando o tamanho da *cache* L2 é maior, a latência de acesso também aumenta.

Nesta dissertação, optou-se por não considerar o efeito da latência de acesso nos diferentes tamanhos de *cache* configurados, garantindo que esse tempo seja o mesmo em qualquer configuração de *cache* L2. Essa latência é parametrizada no *Gem5*, e foi mantida igual para todos os tamanhos de *cache* L2. A análise da influência da latência de acesso à memória será feita em trabalhos futuros.

Finalizadas as configurações e parametrizações do simulador, passa-se a etapa de efetivamente simular as arquiteturas com o processamento das cargas de trabalhos selecionadas. Os parâmetros e configurações serão explicados com mais detalhes no Capítulo 4. Cada configuração de arquitetura é simulada em uma instância de processo, o que permitiu a condução de várias simulações simultaneamente, havendo recursos computacionais disponíveis, sem impactos nos resultados.

Nos estágios finais do processo de exploração do espaço de projeto, ocorre a coleta e avaliação das métricas relacionadas aos objetivos da DSE discutidos na Seção 3.4.1. Ao término das simulações, o *Gem5* produz um arquivo único contendo os dados produzidos em cada simulação. Neste caso, os dados coletados referem-se apenas ao momento em que a aplicação processou, sendo desconsiderado o tempo para início do sistema operacional, por exemplo. Os dados são avaliados, tabulados e discutidos no Capítulo 5. Após a avaliação dos resultados produzidos por cada arquitetura, as alternativas que mais se aproximaram dos objetivos iniciais são destacadas, sendo essas as saídas do processo da DSE, como na Figura 11.

4 PROPOSTA DE ARQUITETURAS SIMULADAS

O objetivo deste capítulo é apresentar, baseada nas discussões realizadas no capítulo anterior, quais são as propostas de arquiteturas para simulação. As arquiteturas propostas são diferenciadas em quantidade de núcleos, tamanhos de *cache* L2 e topologia, com diferentes posicionamentos das *caches* L2 na rede-em-chip.

Para a alteração do número de núcleos, optou-se pela variação em 16, 32, 64 e 128 núcleos de processamento. As quantidades escolhidas aproximam-se de outras arquiteturas *many-core* de mercado, como a *Tilera TILE-Gx* (FLEIG; MATTES; KARL, 2014), com versões de até 72 núcleos, e a *Intel Xeon-Phi* (CHRYSOS; ENGINEER, 2012), com versões de até 61 núcleos. O número de núcleos pode ser facilmente definido no simulador, por meio de parâmetros pré configurados em um arquivo em *Python* chamado *Options.py*. Este arquivo centraliza a maior parte dos parâmetros da simulação, aumentando a flexibilidade de uso da ferramenta.

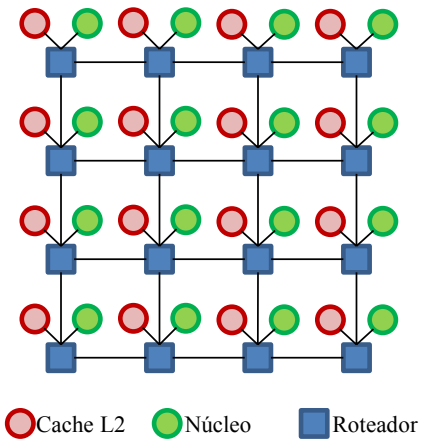
Já para o estudo das *caches* L2, optou-se por simular *caches* com o tamanho de $64kB$ por núcleo, passando por $128kB$, até $256kB$, também por núcleo. Assim sendo, de acordo com a topologia definida e quantidade de núcleos, tem-se arquiteturas com tamanho total de memória *cache* L2 de 1 MB até 32 MB, tamanho este equivalente à memória disponível no MPPA-256 considerando todos os *clusters*. O tamanho de cada unidade de *cache* L2 irá variar de acordo com a topologia configurada. Como o tamanho será definido por quantidade de núcleos, na topologia em que cada roteador se conecta a um núcleo e a uma *cache* L2, tem-se os tamanhos iniciais já mencionados ($64kB$, $128kB$ e $256kB$). Nos casos em que as *caches* são alocadas em roteadores dedicados, elas aumentam em tamanho por 4 vezes, ou seja, tem-se os tamanhos de $256kB$, $512kB$ e $1024kB$. Outros tamanhos por unidade de *cache* são configurados com as topologias do tipo *Cluster*, que serão discutidos a seguir. O tamanho e quantidade de *caches* é configurado por parâmetros do simulador, da mesma forma que a quantidade de núcleos, através do arquivo *Options.py*.

Para configurar a topologia no simulador *Gem5*, é necessário escrever um código, também em *Python*, que represente o formato desejado. A modularidade de componentes do *Gem5*, por orientação a objeto, facilita este trabalho, permitindo que o código se resuma a instanciar componentes, e conectá-los. Para definir uma nova topologia, é necessário criar um arquivo com uma nova classe, que estenda a classe *Topology*. Este arquivo será acionado a partir de outro arquivo, denominado *Ruby.py*, em que são instanciados, de maneira geral, os componentes da arquitetura, como núcleos e *caches*. As instâncias dos componentes (objetos) são repassados à classe da topologia escolhida, e nesta classe, criam-se os objetos que representam os roteadores e *links* de conexão, tanto para a conexão

dos demais componentes com esses roteadores, quanto para a conexão entre os próprios roteadores. Por fim, é necessário criar uma estrutura lógica desses objetos, formando a topologia para uso no simulador.

A partir dos tipos *Mesh*, *Torus* e *Cluster*, definiu-se um conjunto de topologias diversas. Para a topologia *Mesh*, uma organização tradicional foi escolhida como ponto de partida, ilustrada na Figura 12, com formato quadrado, tendo essa organização um roteador para cada núcleo e *cache* L2. A intenção é usar essa configuração como referencia para as demais topologias, verificando as vantagens e desvantagens das outras em relação a ela. Os roteadores em que estão conectadas as *caches* L2 são os mesmos roteadores que possuem interconexão com o sistema de memória principal. Outra característica em todas as topologias, é que todos os núcleos de processamento podem acessar qualquer *cache* presente na rede.

Figura 12 – Topologia Mesh 4x4 com 16 núcleos e 16 *caches* L2 - M0

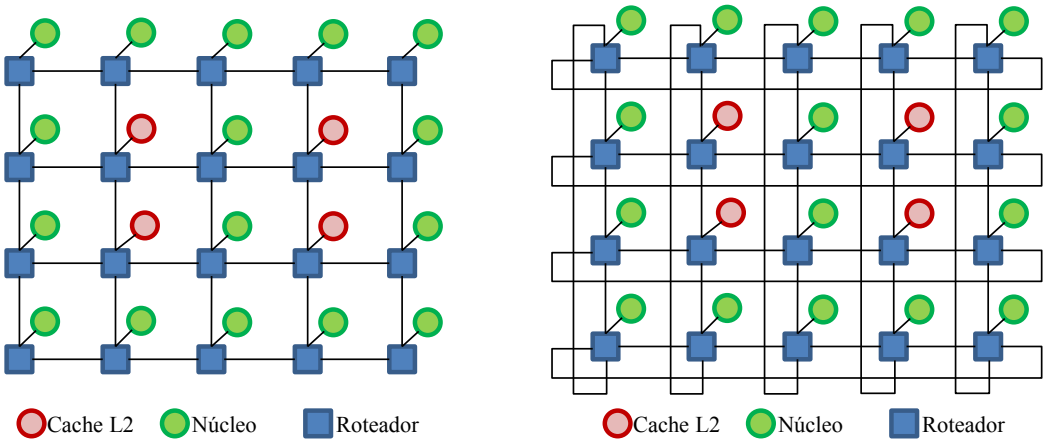


Fonte: Elaborado pelo autor

Outras quatro organizações foram codificadas, nas quais a quantidade de roteadores é igual a soma da quantidade de núcleos e a quantidade de *caches* configuradas. Como o tamanho das *caches* será aumentado em 4 vezes, o número de unidades será reduzido na mesma proporção. Dessa forma, partindo de uma configuração com 16 núcleos, tem-se 4 unidades de *cache*. Logo, a quantidade de roteadores será de 20. O formato terá uma coluna a mais, passando a ser retangular. Essas quatro organizações também foram feitas para o tipo *Torus*, bastando conectar os roteadores das extremidades com novos objetos de *links* nos arquivos de definição de topologias. Com o aumento do número de núcleos, os demais componentes aumentam proporcionalmente. Isto é, em uma configuração com 128 núcleos, tem-se $128/4 = 32$ unidades de *cache* L2, e, logo, 160 roteadores. Essas topologias são ilustradas na Figura 13, Figura 14, Figura 15 e Figura 16.

Figura 13 – Topologias 4x5 com 16 núcleos e 4 *caches* L2 - 1

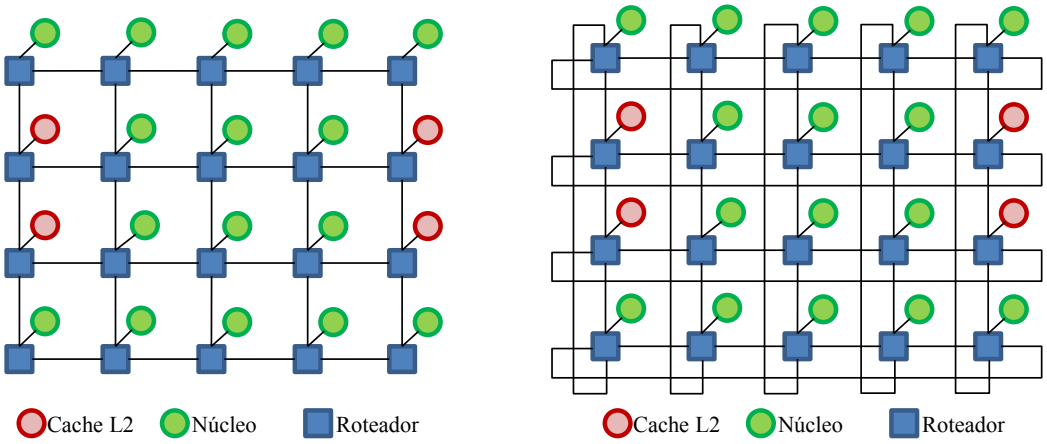
(a) M1 - Mesh **(b) T1 - Torus**



Fonte: Elaborado pelo autor

Figura 14 – Topologias 4x5 com 16 núcleos e 4 *caches* L2 - 2

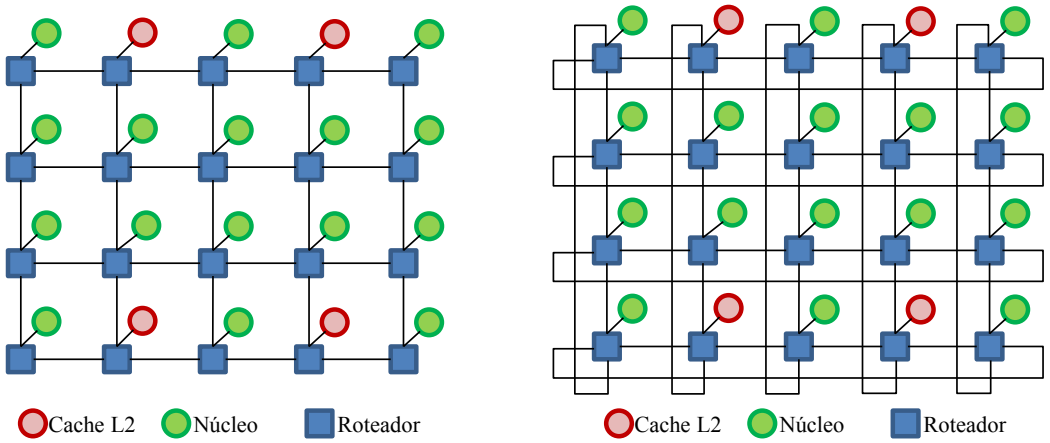
(a) M2 - Mesh **(b) T2 - Torus**



Fonte: Elaborado pelo autor

Figura 15 – Topologias 4x5 com 16 núcleos e 4 *caches* L2 - 3

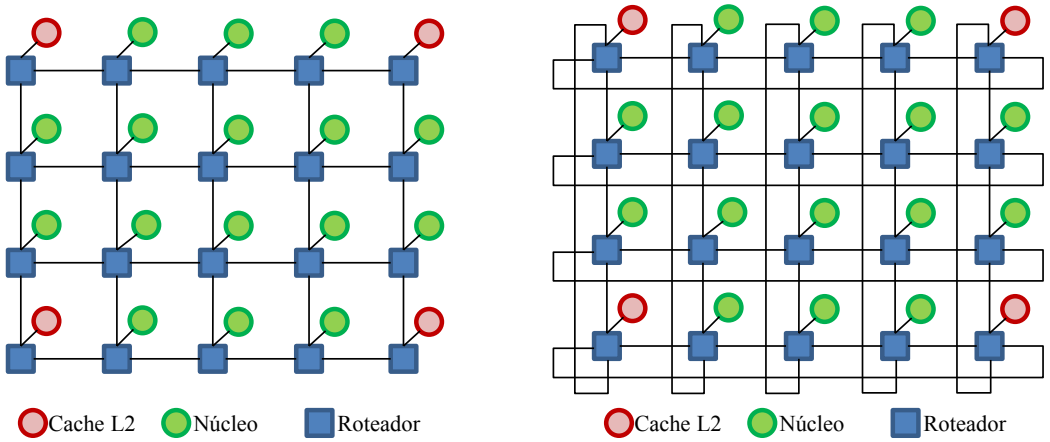
(a) M3 - Mesh **(b) T3 - Torus**



Fonte: Elaborado pelo autor

Figura 16 – Topologias 4x5 com 16 núcleos e 4 *caches* L2 - 4

(a) M4 - Mesh **(b) T4 - Torus**



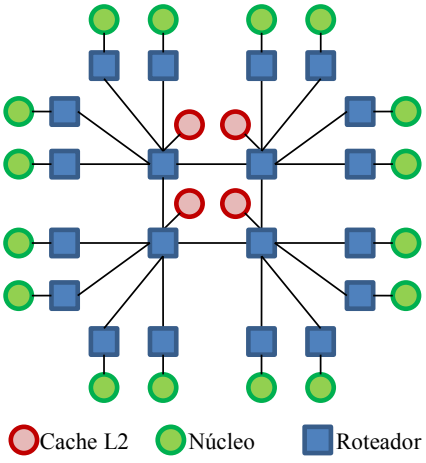
Fonte: Elaborado pelo autor

Para as topologias do tipo *Cluster*, foram configuradas 6 alternativas, em acordo com o número de núcleos. A ideia dessas topologias é ter roteadores centrais, conectados somente às *caches* e demais componentes externos. Esses roteadores estão conectados entre si. Cada um tem, além disso, conectividade com outros roteadores conectados aos núcleos. Assim, para um núcleo comunicar-se com outro, será sempre necessário passar por um dos roteadores centrais. Nos casos de comunicações entre *clusters*, pelo menos 2 roteadores centrais precisam ser acionados. Cada roteador central é conectado a uma *cache* L2 de tamanho equivalente ao número de núcleos agrupados, nos tamanhos por núcleo discutidos anteriormente.

Com 16, 32 e 64 núcleos, foram criadas versões com 4 agrupamentos, codificadas como C4, de, respectivamente, 4, 8 e 16 núcleos cada. Para 64 e 128 núcleos, de maneira alternativa, optou-se por configurar 8 agrupamentos, com o código C8 (8 e 16 núcleos por agrupamento). Uma outra opção com 16 agrupamentos, codificada como C16, foi simulada na versão com 128 núcleos, o que significa ter, em cada agrupamento, 8 núcleos. Um exemplo dessa topologia, com 4 agrupamentos e 16 núcleos, está ilustrado na Figura 17.

É importante ressaltar que a topologia do tipo *Cluster* escala com base em uma topologia do tipo *Mesh*, isto é, quando o número de agrupamentos aumenta, os roteadores centrais formam uma *Mesh*. De forma análoga, o MPPA-256, que também está organizado em *clusters*, escala com base em uma topologia do tipo *Torus*. Nas arquiteturas propostas, a topologia *Cluster* é diferenciada das topologias *Mesh* e *Torus* pelo agrupamento de núcleos para acesso à memórias *cache*.

Figura 17 – Topologia Cluster 2x2 com 16 núcleos e 4 *caches* L2 - C4



Fonte: Elaborado pelo autor

O Quadro 1 apresenta um resumo das topologias produzidas, com a referência ilustrativa para cada uma. É importante relembrar que, além da forma, essas topologias também variam em quantidade de núcleos e tamanho de memória *cache* L2. As topologias foram codificadas para facilitar a discussão no restante do trabalho. Para todas as

topologias, com exceção da de código M0, há uma unidade de *cache* L2 para cada quatro núcleos. Sendo assim, as configurações de M1 a C4, no Quadro 1, possuem 4, 8, 16 e 32 unidades de memória *cache* L2, respectivamente para 16, 32, 64 e 128 núcleos.

Quadro 1 – Resumo básico de topologias

Código	Formato	Figura de referência
M0	*Mesh*	Figura 12
M1	*Mesh*	Figura 13a
M2	*Mesh*	Figura 13b
M3	*Mesh*	Figura 14a
M4	*Mesh*	Figura 14b
T1	*Torus*	Figura 15a
T2	*Torus*	Figura 15b
T3	*Torus*	Figura 16a
T4	*Torus*	Figura 16b
C4	*Cluster*	Figura 17

Fonte: Elaborado pelo autor

Para todas as configurações de topologia, o protocolo de roteamento usado é o XY. O protocolo de coerência de *cache* é o MOESI, uma variação do protocolo MESI, que adiciona um novo estado (*Owned*), que compreende um estado em que o dado em cache está modificado e compartilhado, ao mesmo tempo (SUH; BLOUGH; LEE, 2004).

5 AVALIAÇÃO DAS ARQUITETURAS SIMULADAS

Neste capítulo, são discutidos os resultados obtidos na pesquisa. Esses resultados são baseados nas saídas produzidas pelo simulador *Gem5*, extraindo-se delas as métricas definidas na DSE. Os ganhos ou perdas de desempenho, bem como ganhos ou perdas no consumo de potência ou energia, são apresentados em relação à topologia M0, cujo exemplo está ilustrado na Figura 12. Os gráficos apresentados nas figuras deste capítulo estão todos em escala logarítmica, com base 2. Os ganhos ou redução percentuais apresentados no texto foram calculados em relação aos resultados da topologia M0.

A escalabilidade obtida ou não pelas aplicações é influenciada pela classe escolhida do NPB. Estudos com diferentes classes podem aprofundar o conhecimento sobre a melhor exploração das arquiteturas *many-core* com redes-em-chip. A variação de classes não faz parte do escopo desta dissertação e está presente em propostas de trabalhos futuros. As avaliações de cada aplicação, nesta dissertação, concentram-se apenas na classe *S*.

Esperava-se o funcionamento normal de todas as aplicações durante a simulação. Entretanto, foram apresentados erros de segmentação para algumas delas, quando simuladas em determinadas arquiteturas com certas quantidades de núcleos. São elas as aplicações *CG - Conjugate Gradient* e *IS - Integer Sort* para 64 núcleos, e a aplicação *EP - Embarassingly Parallel* para 128 núcleos. Dessa forma, os resultados para essas aplicações não serão apresentados para essas quantidades. Optou-se por não verificar e depurar o problema, dado que seria necessária avaliação do código dessas aplicações e depuração, ficando, então, esta atividade para trabalhos futuros.

5.1 Resultados para a aplicação CG

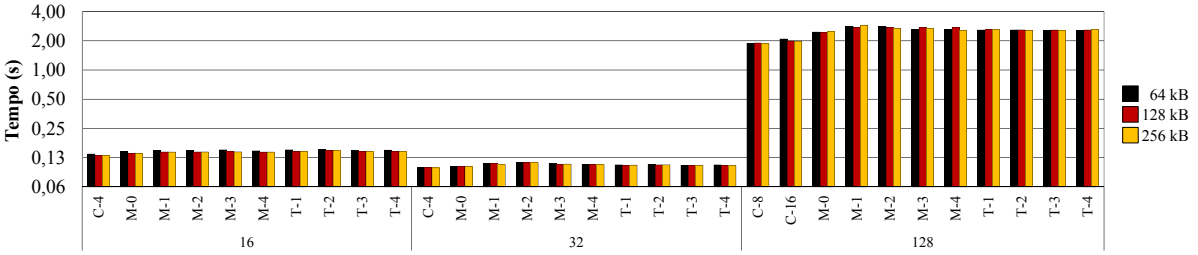
Nesta seção, apresenta-se os resultados para a aplicação CG - *Conjugate Gradient*, que utiliza o método do gradiente conjugado para estimar o menor autovalor de uma matriz simétrica positiva. Esta aplicação caracteriza-se por comunicações irregulares um-para-um que ocorrem entre núcleos próximos (OLIVEIRA, 2012), ou seja, que estão em roteadores diretamente conectados. Os acessos aos dados em memória também são irregulares.

A Figura 18 apresenta o gráfico do tempo de processamento da aplicação CG, juntamente com a Tabela 1, para todas as configurações. Pelo gráfico, nota-se que, para todas as topologias, aumentar de 16 ou 32 para 128 núcleos piorou muito o tempo de processamento. Sem um mapeamento adequado da aplicação, as comunicações entre os núcleos próximos podem não acontecer nos ciclos seguintes, dada a ocupação dos roteadores e links entre esses núcleos naquele momento. Aumentar o número de núcleos piora isso, pois há mais núcleos disponíveis, mas em distâncias maiores. Isso é confirmado

pelo aumento na latência média da rede, ilustrado no gráfico da Figura 19, complementado pelos dados descritos na Tabela 2.

Em linhas gerais, foi possível perceber ganhos no tempo de processamento da aplicação CG, a partir das configurações cuja topologia é a de código M0, mais tradicional. Para 16 núcleos, com a topologia C4, foi possível obter ganhos de $5,25\%$, $4,87\%$ e $5,20\%$ com 64kB, 128kB e 256kB, respectivamente, em relação à topologia M0 também com 16 núcleos. A mesma topologia (C4) foi a que teve melhores resultados também para 32 núcleos, em que obteve-se ganhos para 64kB, 128kB e 256kB de $3,29\%$, $3,15\%$ e $3,56\%$, nesta ordem, em relação à topologia M0. Para 128 núcleos, a topologia do tipo *Cluster* C8 destacou-se com ganhos maiores em relação à M0: um tempo $23,73\%$ melhor para 64kB de *cache* L2, $22,16\%$ para 128kB e $25,00\%$ para 256kB.

Figura 18 – Aplicação CG - Tempo de processamento da aplicação



Núcleos de processamento

Fonte: Dados da pesquisa

Tabela 1 – Aplicação CG - Tempo de processamento da aplicação (s)

Topologia	***Cache* L2 por núcleo**	**Núcleos**		
		16	32	128
C4	64 kB	0,1362	0,0986	-
	128 kB	0,1314	0,0985	-
	256 kB	0,1307	0,0981	-
C8	64 kB	-	-	1,8751
	128 kB	-	-	1,8990
	256 kB	-	-	1,8606
C16	64 kB	-	-	2,0857
	128 kB	-	-	1,9876
	256 kB	-	-	1,9826
M0	64 kB	0,1438	0,1019	2,4585
	128 kB	0,1381	0,1017	2,4396
	256 kB	0,1379	0,1017	2,4809

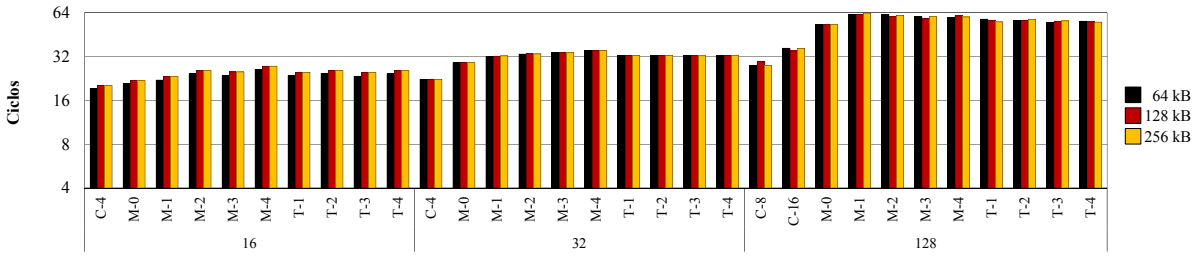
Continua

Conclusão

Topologia	*Cache* L2 por núcleo	Núcleos		
		16	32	128
M1	64 kB	0,1468	0,1095	2,8259
	128 kB	0,1428	0,1090	2,7578
	256 kB	0,1425	0,1062	2,8982
M2	64 kB	0,1470	0,1111	2,7957
	128 kB	0,1430	0,1110	2,7551
	256 kB	0,1428	0,1110	2,6787
M3	64 kB	0,1502	0,1095	2,6294
	128 kB	0,1434	0,1070	2,7282
	256 kB	0,1432	0,1073	2,6773
M4	64 kB	0,1467	0,1060	2,6236
	128 kB	0,1427	0,1059	2,7239
	256 kB	0,1425	0,1056	2,5409
T1	64 kB	0,1501	0,1052	2,5795
	128 kB	0,1453	0,1043	2,5956
	256 kB	0,1453	0,1046	2,6051
T2	64 kB	0,1523	0,1057	2,5865
	128 kB	0,1482	0,1053	2,5844
	256 kB	0,1481	0,1053	2,5664
T3	64 kB	0,1488	0,1039	2,5351
	128 kB	0,1446	0,1038	2,5811
	256 kB	0,1442	0,1036	2,5364
T4	64 kB	0,1493	0,1053	2,5508
	128 kB	0,1452	0,1045	2,5820
	256 kB	0,1451	0,1038	2,6032
	Melhor resultado	0,1307	0,0981	1,8606
	Pior resultado	0,1523	0,1111	2,8982

Fonte: Dados da pesquisa

Figura 19 – Aplicação CG - Latência de rede



Núcleos de processamento

Fonte: Dados da pesquisa

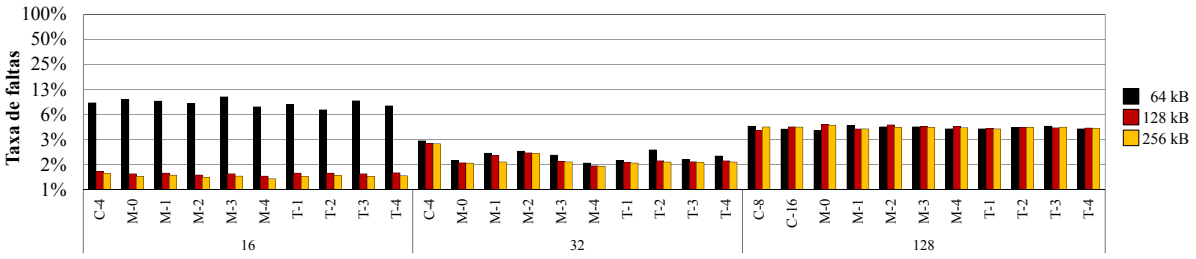
Tabela 2 – Aplicação CG - Latência de rede (ciclos)

Topologia	*Cache* L2 por núcleo	Núcleos		
		16	32	128
C4	64 kB	19,34	22,34	-
	128 kB	20,27	22,37	-
	256 kB	20,26	22,35	-
C8	64 kB	-	-	27,76
	128 kB	-	-	29,46
	256 kB	-	-	27,69
C16	64 kB	-	-	36,55
	128 kB	-	-	34,92
	256 kB	-	-	36,46
M0	64 kB	20,76	28,98	53,06
	128 kB	21,89	29,01	53,13
	256 kB	21,91	29,01	52,94
M1	64 kB	22,06	32,13	62,47
	128 kB	23,29	32,17	62,37
	256 kB	23,30	32,29	63,67
M2	64 kB	24,43	33,28	61,91
	128 kB	25,72	33,37	60,14
	256 kB	25,74	33,32	61,48
M3	64 kB	23,56	33,88	60,05
	128 kB	25,07	33,96	58,56
	256 kB	25,08	33,97	60,18
M4	64 kB	26,13	35,10	59,00
	128 kB	27,48	35,14	60,77
	256 kB	27,50	35,15	59,79
T1	64 kB	23,73	32,46	57,83
	128 kB	24,91	32,49	56,27
	256 kB	24,94	32,50	55,34
T2	64 kB	24,54	32,32	56,73
	128 kB	25,62	32,48	56,36
	256 kB	25,64	32,51	57,46
T3	64 kB	23,44	32,31	55,00
	128 kB	24,82	32,35	55,67
	256 kB	24,84	32,35	56,20
T4	64 kB	24,38	32,30	55,99
	128 kB	25,57	32,38	55,30
	256 kB	25,59	32,39	55,19
	Melhor resultado	19,34	22,34	27,69
	Pior resultado	27,50	35,15	63,67

Fonte: Dados da pesquisa

A taxa de faltas na memória *cache* L2 é mostrada no gráfico da Figura 20, complementado pela Tabela 3. Essa taxa mostrou-se elevada nas configurações com 16 núcleos e 64kB por núcleo, o que justifica-se pela quantidade de *cache* L2 disponível (um total de 1MB), que é menor que nas demais configurações. Ter 2MB de *cache* L2, ou mais do que isso, é suficiente para a aplicação CG ter uma redução nessa métrica, considerando também sua granularidade de dados. Portanto, ocorreu pouca alteração nessa métrica quando houve a mudança de 128kB para 256kB. Por outro lado, à medida em que o número de núcleos aumenta, a taxa de faltas também aumenta. O tipo de comunicação um-para-um é prejudicado nesse sentido, já que a latência média foi alta. Essa estratégia de comunicação fez com que dados ficassem 'em espera' na memória, aguardando a conclusão de alguma comunicação unidirecional entre dois núcleos. Logo, a concorrência de outros núcleos pelos recursos de memória compartilhados fez com que esses dados 'em espera' fossem retirados da memória para inclusão de outros. O uso de muito mais núcleos então precisa levar em consideração estratégias de mapeamento de processos, remodelando a aplicação e as distâncias entre os núcleos que se comunicam, o que é sugerido como trabalho futuro.

Figura 20 – Aplicação CG - Taxa de faltas na L2



Núcleos de processamento

Fonte: Dados da pesquisa

Tabela 3 – Aplicação CG - Taxa de faltas na *cache* L2

Topologia	*Cache* L2 por núcleo	Núcleos		
		16	32	128
C4	64 kB	8,64%	3,01%	-
	128 kB	1,29%	2,80%	-
	256 kB	1,22%	2,78%	-
C8	64 kB	-	-	4,53%
	128 kB	-	-	4,04%
	256 kB	-	-	4,42%
C16	64 kB	-	-	4,17%
	128 kB	-	-	4,44%
	256 kB	-	-	4,40%

Continua

Fonte: Dados da pesquisa

Conclusão

Topologia	*Cache* L2 por núcleo	Núcleos		
		16	32	128
M0	64 kB	9,54%	1,76%	4,01%
	128 kB	1,21%	1,65%	4,76%
	256 kB	1,13%	1,62%	4,65%
M1	64 kB	9,04%	2,14%	4,61%
	128 kB	1,23%	2,02%	4,18%
	256 kB	1,16%	1,68%	4,19%
M2	64 kB	8,52%	2,27%	4,45%
	128 kB	1,17%	2,16%	4,69%
	256 kB	1,09%	2,14%	4,35%
M3	64 kB	10,19%	2,03%	4,45%
	128 kB	1,21%	1,70%	4,51%
	256 kB	1,14%	1,68%	4,38%
M4	64 kB	7,76%	1,63%	4,20%
	128 kB	1,14%	1,52%	4,50%
	256 kB	1,05%	1,49%	4,33%
T1	64 kB	8,25%	1,76%	4,22%
	128 kB	1,23%	1,65%	4,26%
	256 kB	1,12%	1,63%	4,18%
T2	64 kB	7,14%	2,34%	4,38%
	128 kB	1,23%	1,74%	4,37%
	256 kB	1,15%	1,67%	4,35%
T3	64 kB	9,17%	1,80%	4,52%
	128 kB	1,21%	1,69%	4,34%
	256 kB	1,13%	1,65%	4,40%
T4	64 kB	7,96%	1,98%	4,22%
	128 kB	1,24%	1,73%	4,30%
	256 kB	1,15%	1,67%	4,27%
	Melhor resultado	1,05%	1,49%	4,01%
	Pior resultado	10,19%	3,01%	4,76%

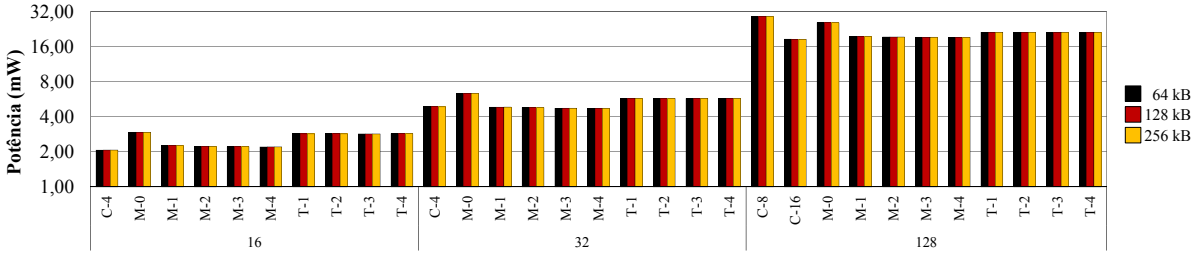
Fonte: Dados da pesquisa

Com relação à potência consumida pela rede, os resultados podem ser verificados no gráfico da Figura 21 e na Tabela 4. Associa-se os resultados de potência à variação nos componentes da rede-em-chip. O primeiro fator que confirma isso, é o aumento de núcleos. Aumentar núcleos significa aumentar roteadores e links, que são mais componentes consumidores de energia dentro do chip. Os resultados para as topologias de código M0, dada como referência inicial, são maiores, pois mais unidades de cache L2 estão presentes na rede-em-chip, o que significa mais links interconectando essas *caches* com os roteadores da rede. As alterações no consumo em razão da mudança no tamanho da cache L2 por núcleo está mais atrelado à alterações no tempo de processamento das aplicações, do que aos componentes propriamente ditos. A rede não sofre influência direta, em termos de

consumo energético, do tamanho das memórias, mas sim da quantidade, como no caso dos links. Em um trabalho posterior, a avaliação do consumo energético das memórias pode ser realizado.

Os ganhos em relação à potência consumida mostraram a aplicabilidade da topologia C4 em relação à M0, ambas com 16 núcleos. Neste caso, foram obtidas reduções no consumo de potência na faixa de $29,70\%$ a $29,80\%$, para todos os tamanhos de *cache* L2 por núcleo, na topologia C4 em relação à M0. Para 32 núcleos, a topologia que se destacou foi a M4, com reduções no consumo de potência da rede-em-chip próximas de $25,76\%$ também em relação à M0. Com 128 núcleos, a topologia do tipo *Cluster* volta a ter destaque. Em relação à M0, a topologia C16 foi a que proporcionou uma parcela maior de economia no consumo, com reduções de até $27,72\%$.

Figura 21 – Aplicação CG - Potência consumida pela rede



Núcleos de processamento

Fonte: Dados da pesquisa

Tabela 4 – Aplicação CG - Potência consumida pela rede (mW)

Topologia	*Cache* L2 por núcleo	Núcleos		
		16	32	128
C4	64 kB	2,0682	4,8688	-
	128 kB	2,0700	4,8678	-
	256 kB	2,0689	4,8668	-
C8	64 kB	-	-	28,9843
	128 kB	-	-	29,0350
	256 kB	-	-	28,9911
C16	64 kB	-	-	18,5332
	128 kB	-	-	18,5160
	256 kB	-	-	18,5273
M0	64 kB	2,9421	6,3654	25,6411
	128 kB	2,9471	6,3654	25,6148
	256 kB	2,9471	6,3647	25,6188

Continua

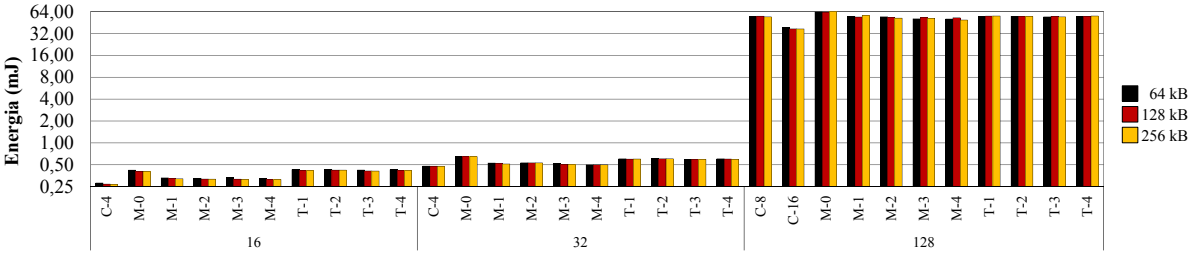
Conclusão

Topologia	*Cache* L2 por núcleo	Núcleos		
		16	32	128
M1	64 kB	2,2549	4,8308	19,4058
	128 kB	2,2551	4,8317	19,4253
	256 kB	2,2551	4,8412	19,4198
M2	64 kB	2,2271	4,7937	19,3374
	128 kB	2,2272	4,7946	19,3320
	256 kB	2,2270	4,7924	19,3523
M3	64 kB	2,2113	4,7277	19,2746
	128 kB	2,2142	4,7358	19,2641
	256 kB	2,2141	4,7338	19,2733
M4	64 kB	2,2010	4,7257	19,2259
	128 kB	2,2028	4,7249	19,2155
	256 kB	2,2027	4,7255	19,2236
T1	64 kB	2,8510	5,7458	21,2694
	128 kB	2,8519	5,7482	21,2621
	256 kB	2,8528	5,7464	21,2642
T2	64 kB	2,8532	5,7454	21,2631
	128 kB	2,8532	5,7432	21,2607
	256 kB	2,8530	5,7449	21,2662
T3	64 kB	2,8481	5,7400	21,2552
	128 kB	2,8476	5,7396	21,2618
	256 kB	2,8477	5,7388	21,2626
T4	64 kB	2,8649	5,7423	21,2771
	128 kB	2,8653	5,7442	21,2720
	256 kB	2,8650	5,7452	21,2706
	Melhor resultado	2,0682	4,7249	18,5160
	Pior resultado	2,9471	6,3654	29,0350

Fonte: Dados da pesquisa

Pelo gráfico da Figura 22 e pela Tabela 5, é possível constatar os melhores resultados de consumo energético durante a simulação da carga CG para a topologia C4, quando com 16 e 32 núcleos ($0,2704\ mJ$ e $0,4772\ mJ$), ambos para a configuração com 256kB de *cache* L2 por núcleo. Já para 128 núcleos, o melhor resultado foi de $36,7323\ mJ$, obtido com a topologia C16, também com 256kB de *cache* L2 por núcleo. Nota-se, então, que a aplicação CG tem ganhos de performance quando executada em uma topologia do tipo *Cluster*, quando comparada com as topologias tradicionais *Mesh* e *Torus*. O fato de ser uma aplicação com padrões de comunicação um-para-um é favorecido pela redução de saltos em uma comunicação, já que a topologia *Cluster* encurta a distância entre os vários núcleos. Esse mesmo comportamento foi observado nos trabalhos desenvolvidos por Oliveira (2012) e Avelar (2014).

Figura 22 – Aplicação CG - Energia consumida pela rede



Núcleos de processamento

Fonte: Dados da pesquisa

O agrupamento de núcleos na topologia *Cluster*, quando comparada às topologias *Mesh* ou *Torus*, reduziu a quantidade de links para interconexão dos roteadores e consequentemente o consumo de potência. Além disso, com as distâncias menores, a atividade de roteamento é menor, reduzindo o uso dos roteadores, no somatório total, algo que reduz muito o consumo de energia da rede-em-chip (KIM et al., 2011). Essa aplicação é caracterizada, ainda, por pacotes de tamanho grande, o que justifica melhores resultados com tamanhos maiores de *cache* L2. Há mais espaço na memória, logo, mais dados podem ocupá-la. As transmissões de dados entre os núcleos e as *caches* são feitas em menor número, reduzindo a atividade e, consequentemente, o consumo energético.

Tabela 5 – Aplicação CG - Energia consumida pela rede (mJ)

Topologia	***Cache* L2 por núcleo**	**Núcleos**		
		16	32	128
C4	64 kB	0,2817	0,4799	-
	128 kB	0,2719	0,4794	-
	256 kB	0,2704	0,4772	-
C8	64 kB	-	-	54,3491
	128 kB	-	-	55,1378
	256 kB	-	-	53,9419
C16	64 kB	-	-	38,6543
	128 kB	-	-	36,8026
	256 kB	-	-	36,7323
M0	64 kB	0,4230	0,6488	63,0382
	128 kB	0,4070	0,6473	62,4892
	256 kB	0,4063	0,6471	63,5570
M1	64 kB	0,3310	0,5291	54,8384
	128 kB	0,3221	0,5269	53,5711
	256 kB	0,3213	0,5139	56,2824
M2	64 kB	0,3274	0,5324	54,0624
	128 kB	0,3184	0,5321	53,2608
	256 kB	0,3181	0,5320	51,8387

Continua

Conclusão

Topologia	*Cache* L2 por núcleo	Núcleos		
		16	32	128
M3	64 kB	0,3322	0,5176	50,6815
	128 kB	0,3176	0,5069	52,5570
	256 kB	0,3171	0,5079	51,6001
M4	64 kB	0,3228	0,5008	50,4415
	128 kB	0,3143	0,5005	52,3414
	256 kB	0,3139	0,4992	48,8445
T1	64 kB	0,4281	0,6044	54,8638
	128 kB	0,4145	0,5997	55,1889
	256 kB	0,4145	0,6012	55,3944
T2	64 kB	0,4345	0,6070	54,9965
	128 kB	0,4228	0,6046	54,9467
	256 kB	0,4226	0,6050	54,5776
T3	64 kB	0,4239	0,5967	53,8845
	128 kB	0,4117	0,5959	54,8778
	256 kB	0,4107	0,5944	53,9301
T4	64 kB	0,4278	0,6045	54,2745
	128 kB	0,4159	0,6002	54,9241
	256 kB	0,4156	0,5961	55,3720
	Melhor resultado	0,2704	0,4772	36,7323
	Pior resultado	0,4345	0,6488	63,5570

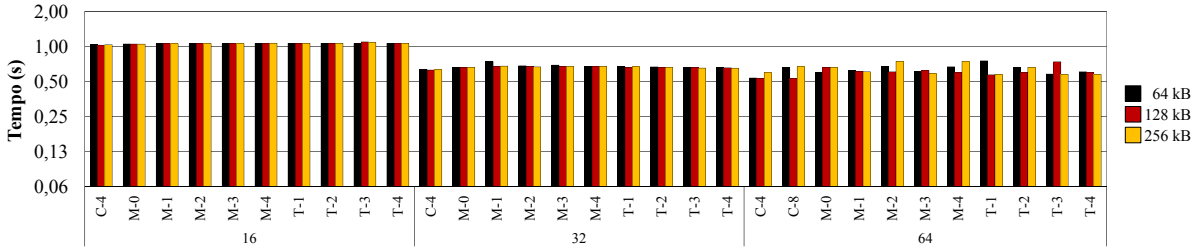
Fonte: Dados da pesquisa

5.2 Resultados para a aplicação EP

A aplicação EP - *Embarassingly Parallel* produz um conjunto de pares gaussianos aleatórios e independentes e combina as somas desses pares usando o método polar de *Marsaglia*. Esta aplicação não possui quantidades de comunicações ou acessos à memória significantes, sendo caracterizada por comunicações com pacotes pequenos, trabalhando com dados de granularidade menor. Isto é, trabalha com dados de tamanho inferior em relação às demais aplicações, prevalecendo nela o tipo de comunicação muitos-para-muitos, ou *broadcast*. Para a aplicação EP, não serão apresentados resultados com a topologia C16, uma vez que essa topologia somente foi simulada com 128 núcleos, e o processamento da aplicação não foi concluído para essa quantidade.

O gráfico da Figura 23 mostra os tempos de processamento da aplicação EP, descritos também na Tabela 6. Para todas as topologias definidas, aumentar o número de núcleos para 64 apresentou poucas vantagens. Em alguns casos, foram observados resultados de tempo piores do que com um número menor de núcleos.

Figura 23 – Aplicação EP - Tempo de processamento da aplicação



Núcleos de processamento

Fonte: Dados da pesquisa

A topologia *Cluster* C4 apresentou bons resultados no tempo de processamento da aplicação, em relação à topologia M0. Quando foram usados 16 núcleos e 64kB de memória *cache* L2, houve um ganho de $1,29\%$ em relação à topologia M0. O mesmo ocorreu para 128kB e 256kB, com $2,15\%$ e $1,33\%$ de redução no tempo, respectivamente, comparando-se também com a topologia M0. Para 32 núcleos, alterar da topologia M0 para C4 provocou $4,56\%$ de redução no tempo de processamento na configuração com 64kB, $4,97\%$ com 128kB e $4,49\%$ para 256kB. Com 64 núcleos, também se destacou a topologia C4, com ganhos de $10,66\%$ e $19,20\%$ com 64kB e 128kB, respectivamente, em relação à M0. Entretanto, quando usou-se 256kB de *cache* por núcleo, a topologia com ganhos mais consideráveis foi a do tipo *Torus* T4, com $13,08\%$ de redução no tempo de processamento da aplicação, quando comparada à topologia do tipo *Mesh* M0.

Tabela 6 – Aplicação EP - Tempo de processamento da aplicação (s)

Topologia	***Cache* L2 por núcleo**	**Núcleos**		
		16	32	64
C4	64 kB	1,0372	0,6312	0,5368
	128 kB	1,0263	0,6281	0,5329
	256 kB	1,0343	0,6316	0,5966
C8	64 kB	-	-	0,6636
	128 kB	-	-	0,5344
	256 kB	-	-	0,6724
M0	64 kB	1,0508	0,6614	0,6008
	128 kB	1,0488	0,6610	0,6595
	256 kB	1,0482	0,6613	0,6624
M1	64 kB	1,0655	0,7417	0,6189
	128 kB	1,0575	0,6744	0,6123
	256 kB	1,0577	0,6780	0,6063

Continua

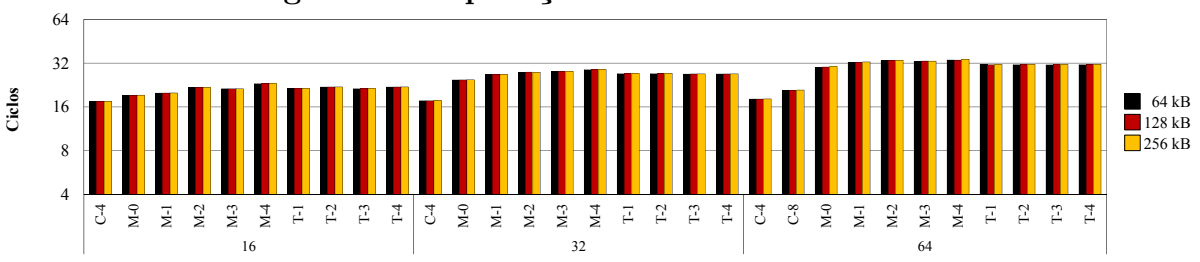
Conclusão

Topologia	*Cache* L2 por núcleo	Núcleos		
		16	32	64
M3	64 kB	1,0630	0,6927	0,6076
	128 kB	1,0652	0,6742	0,6192
	256 kB	1,0577	0,6739	0,5864
M4	64 kB	1,0637	0,6755	0,6690
	128 kB	1,0594	0,6754	0,6001
	256 kB	1,0671	0,6751	0,7431
T1	64 kB	1,0656	0,6700	0,7530
	128 kB	1,0665	0,6628	0,5711
	256 kB	1,0655	0,6704	0,5765
T2	64 kB	1,0658	0,6683	0,6572
	128 kB	1,0656	0,6626	0,5965
	256 kB	1,0655	0,6637	0,6639
T3	64 kB	1,0603	0,6599	0,5795
	128 kB	1,0941	0,6599	0,7391
	256 kB	1,0823	0,6524	0,5762
T4	64 kB	1,0660	0,6587	0,6068
	128 kB	1,0656	0,6553	0,5995
	256 kB	1,0654	0,6519	0,5758
	Melhor resultado	1,0263	0,6281	0,5329
	Pior resultado	1,0941	0,7417	0,7530

Fonte: Dados da pesquisa

A granularidade menor da aplicação EP faz com que várias comunicações sejam realizadas entre os núcleos durante a execução do programa, gerando mais acessos à rede-em-chip. Esse tipo de comunicação em uma rede-em-chip com vários núcleos produz gargalos no uso da rede, que acaba inundada por muitos pequenos pacotes das várias comunicações, que atrasam a computação. Por outro lado, as comunicações de pacotes menores são rápidas, o que justifica os valores menores de latência da rede para a aplicação EP em relação à aplicação CG - *Conjugate Gradient*. Esses valores são apresentados no gráfico da Figura 24 e na Tabela 7.

Figura 24 – Aplicação EP - Latência de rede



Núcleos de processamento

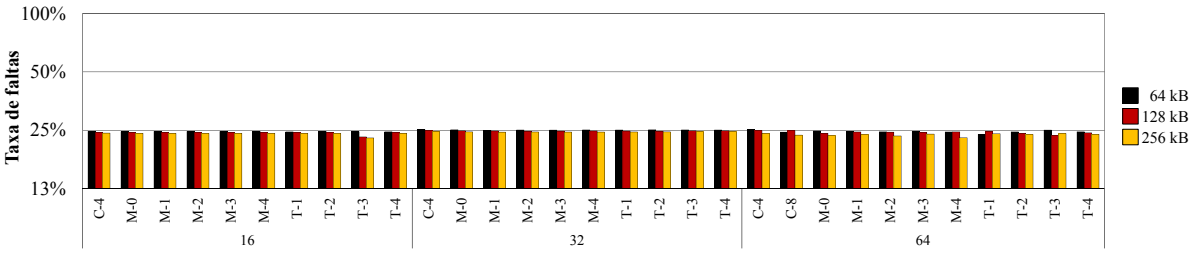
Fonte: Dados da pesquisa

Tabela 7 – Aplicação EP - Latência de rede (ciclos)

Topologia	*Cache* L2 por núcleo	Núcleos		
		16	32	64
C4	64 kB	17,49	17,65	18,11
	128 kB	17,50	17,67	18,12
	256 kB	17,52	17,70	18,17
C8	64 kB	-	-	20,79
	128 kB	-	-	20,77
	256 kB	-	-	20,89
M0	64 kB	19,21	24,50	29,94
	128 kB	19,24	24,52	30,07
	256 kB	19,26	24,58	30,28
M1	64 kB	19,90	26,75	32,41
	128 kB	19,94	26,71	32,43
	256 kB	19,96	26,83	32,61
M2	64 kB	21,78	27,60	33,31
	128 kB	21,80	27,72	33,34
	256 kB	21,83	27,71	33,48
M3	64 kB	21,31	28,02	32,83
	128 kB	21,31	28,10	33,03
	256 kB	21,35	28,16	33,02
M4	64 kB	23,12	28,85	33,57
	128 kB	23,16	28,90	33,55
	256 kB	23,19	28,97	34,04
T1	64 kB	21,44	27,10	31,42
	128 kB	21,46	27,17	31,20
	256 kB	21,49	27,21	31,31
T2	64 kB	21,90	27,08	31,19
	128 kB	21,93	27,14	31,37
	256 kB	21,97	27,21	31,36
T3	64 kB	21,33	26,89	31,09
	128 kB	21,40	26,95	31,43
	256 kB	21,44	26,98	31,30
T4	64 kB	21,89	26,90	31,25
	128 kB	21,93	26,95	31,35
	256 kB	21,95	26,99	31,34
	Melhor resultado	17,49	17,65	18,11
	Pior resultado	23,19	28,97	34,04

Fonte: Dados da pesquisa

Figura 25 – Aplicação EP - Taxa de faltas na L2



Núcleos de processamento

Fonte: Dados da pesquisa

As taxas de faltas apresentadas no gráfico da Figura 25, e também na Tabela 8, mantiveram-se em um patamar entre $22,83\%$ a $25,19\%$ para todas as topologias, quantidades de núcleos e tamanhos de *cache* L2. Em geral, as *caches* maiores apresentam os melhores resultados, ainda que as alterações observadas sejam pouco significativas quando há mudança no tamanho da *cache*. O alto nível de paralelismo dessa aplicação, graças à sua granularidade de dados fina, gera um trabalho muito intenso de redução no final da região paralela. Nesta etapa, muitos valores precisam ser consolidados, e um uso em excesso da *cache* com troca de dados pode ter ocorrido, repondo valores que foram substituídos durante computações em trechos anteriores do programa. Ocorre então o número razoável de faltas na *cache* L2.

Tabela 8 – Aplicação EP - Taxa de faltas na *cache* L2

Topologia	***Cache* L2 por núcleo**	**Núcleos**		
		16	32	64
C4	64 kB	24,69%	25,18%	25,19%
	128 kB	24,42%	24,95%	24,93%
	256 kB	24,19%	24,61%	24,16%
C8	64 kB	-	-	24,48%
	128 kB	-	-	24,89%
	256 kB	-	-	23,63%
M0	64 kB	24,59%	25,17%	24,82%
	128 kB	24,38%	24,86%	24,13%
	256 kB	24,16%	24,54%	23,48%
M1	64 kB	24,66%	24,98%	24,68%
	128 kB	24,37%	24,81%	24,51%
	256 kB	24,12%	24,47%	23,79%
M2	64 kB	24,66%	25,09%	24,56%
	128 kB	24,37%	24,75%	24,47%
	256 kB	24,11%	24,51%	23,37%

Continua

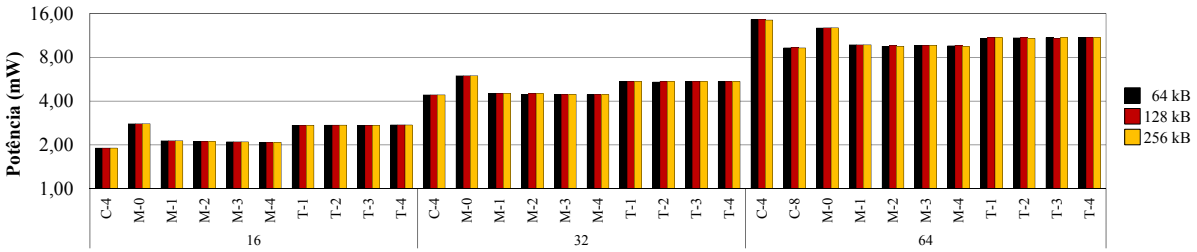
Conclusão

Topologia	*Cache* L2 por núcleo	Núcleos		
		16	32	64
M3	64 kB	24,64%	25,08%	24,75%
	128 kB	24,37%	24,76%	24,39%
	256 kB	24,12%	24,49%	23,89%
M4	64 kB	24,69%	25,11%	24,52%
	128 kB	24,38%	24,82%	24,54%
	256 kB	24,14%	24,49%	22,87%
T1	64 kB	24,53%	25,10%	23,81%
	128 kB	24,35%	24,80%	24,60%
	256 kB	24,11%	24,51%	23,99%
T2	64 kB	24,70%	25,15%	24,53%
	128 kB	24,36%	24,74%	24,16%
	256 kB	24,12%	24,54%	23,80%
T3	64 kB	24,58%	25,11%	25,07%
	128 kB	23,12%	24,85%	23,54%
	256 kB	22,83%	24,59%	24,05%
T4	64 kB	24,56%	25,08%	24,57%
	128 kB	24,36%	24,85%	24,26%
	256 kB	24,12%	24,57%	23,87%
	Melhor resultado	22,83%	24,47%	22,87%
	Pior resultado	24,70%	25,18%	25,19%

Fonte: Dados da pesquisa

Com relação à potência consumida pela rede, novamente percebe-se, pelo gráfico da Figura 26 e pelos dados na Tabela 9, a associação dos resultados com as alterações nos componentes da rede-em-chip. As topologias com mais roteadores (*Cluster*) e links (*Torus* e M0) mostram um consumo energético da rede maior em relação às demais configurações. Entretanto, avaliando-se com uma mesma quantidade de núcleos, foram notados ganhos das topologias *Cluster*. É o caso da topologia C4, com reduções no consumo de potência da rede de até $31,87\%$, com 16 núcleos, e de até $26,05\%$ para 32 núcleos, se comparada à topologia M0. No caso das configurações com 64 núcleos, a topologia que teve maior redução no consumo em relação à M0, foi a C8, com $27,00\%$ de redução para 64kB de *cache* L2 por núcleo, $26,41\%$ de redução para 128kB e, por fim, $27,19\%$ de redução no consumo quando o tamanho da *cache* é de 256kB por núcleo.

Figura 26 – Aplicação EP - Potência consumida pela rede



Núcleos de processamento

Fonte: Dados da pesquisa

Tabela 9 – Aplicação EP - Potência consumida pela rede (mW)

Topologia	*Cache* L2 por núcleo	Núcleos		
		16	32	64
C4	64 kB	1,9034	4,4102	14,5035
	128 kB	1,9030	4,4107	14,5084
	256 kB	1,9021	4,4088	14,4177
C8	64 kB	-	-	9,2806
	128 kB	-	-	9,3789
	256 kB	-	-	9,2765
M0	64 kB	2,7929	5,9622	12,7131
	128 kB	2,7922	5,9618	12,7442
	256 kB	2,7921	5,9615	12,7404
M1	64 kB	2,1350	4,5117	9,7421
	128 kB	2,1346	4,5243	9,7357
	256 kB	2,1345	4,5234	9,7367
M2	64 kB	2,1035	4,4500	9,5312
	128 kB	2,1027	4,4922	9,6739
	256 kB	2,1027	4,4934	9,5115
M3	64 kB	2,1001	4,4445	9,6720
	128 kB	2,0991	4,4479	9,6574
	256 kB	2,0993	4,4477	9,6725
M4	64 kB	2,0827	4,4298	9,5840
	128 kB	2,0824	4,4295	9,6124
	256 kB	2,0819	4,4293	9,4531
T1	64 kB	2,7315	5,4462	10,7496
	128 kB	2,7306	5,4474	10,9059
	256 kB	2,7306	5,4457	10,8988
T2	64 kB	2,7334	5,4090	10,8707
	128 kB	2,7325	5,4474	10,8945
	256 kB	2,7324	5,4473	10,7698

Continua

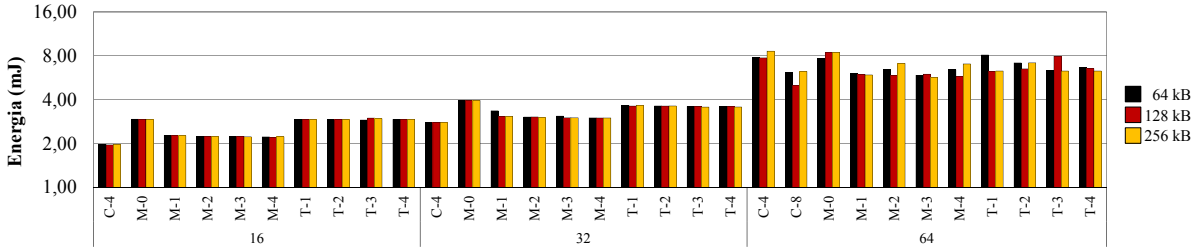
Conclusão

Topologia	*Cache* L2 por núcleo	Núcleos		
		16	32	64
T3	64 kB	2,7312	5,4463	10,9073
	128 kB	2,7297	5,4460	10,7516
	256 kB	2,7302	5,4470	10,8997
T4	64 kB	2,7435	5,4562	10,9043
	128 kB	2,7426	5,4565	10,9006
	256 kB	2,7424	5,4567	10,9093
	Melhor resultado	1,9021	4,4088	9,2765
	Pior resultado	2,7929	5,9622	14,5084

Fonte: Dados da pesquisa

Com a aplicação EP, os melhores resultados também são para as topologias do tipo *Cluster*, como pode ser notado na Tabela 10 e no gráfico da Figura 27. Nas versões com 16 e 32 núcleos, a topologia C4 foi a melhor dentre todas, ambas com 128kB de *cache* L2 por núcleo. Os resultados, nesses casos, foram de $1,9531\ mJ$ e $2,7704\ mJ$, nessa ordem. Com 64 núcleos, a configuração com mais agrupamentos foi melhor, sendo essa configuração a com a topologia C8 e também 128kB, que apresentou $5,0123\ mJ$ de consumo de energia.

Figura 27 – Aplicação EP - Energia consumida pela rede



Núcleos de processamento

Fonte: Dados da pesquisa

Tabela 10 – Aplicação EP - Energia consumida pela rede (mJ)

Topologia	*Cache* L2 por núcleo	Núcleos		
		16	32	64
C4	64 kB	1,9743	2,7839	7,7849
	128 kB	1,9531	2,7704	7,7310
	256 kB	1,9673	2,7846	8,6016
C8	64 kB	-	-	6,1590
	128 kB	-	-	5,0123
	256 kB	-	-	6,2371

Continua

Conclusão

Topologia	*Cache* L2 por núcleo	Núcleos		
		16	32	64
M0	64 kB	2,9348	3,9434	7,6382
	128 kB	2,9286	3,9406	8,4047
	256 kB	2,9267	3,9424	8,4389
M1	64 kB	2,2748	3,3464	6,0289
	128 kB	2,2573	3,0512	5,9610
	256 kB	2,2576	3,0666	5,9035
M2	64 kB	2,2363	3,0357	6,4379
	128 kB	2,2343	3,0395	5,8593
	256 kB	2,2268	3,0029	7,0890
M3	64 kB	2,2325	3,0789	5,8763
	128 kB	2,2359	2,9988	5,9798
	256 kB	2,2203	2,9972	5,6721
M4	64 kB	2,2154	2,9924	6,4118
	128 kB	2,2061	2,9916	5,7683
	256 kB	2,2215	2,9901	7,0242
T1	64 kB	2,9107	3,6490	8,0944
	128 kB	2,9120	3,6107	6,2285
	256 kB	2,9094	3,6506	6,2835
T2	64 kB	2,9131	3,6148	7,1441
	128 kB	2,9118	3,6094	6,4988
	256 kB	2,9115	3,6152	7,1504
T3	64 kB	2,8959	3,5941	6,3209
	128 kB	2,9867	3,5937	7,9466
	256 kB	2,9549	3,5537	6,2808
T4	64 kB	2,9246	3,5941	6,6163
	128 kB	2,9226	3,5754	6,5349
	256 kB	2,9219	3,5572	6,2810
	Melhor resultado	1,9531	2,7704	5,0123
	Pior resultado	2,9867	3,9434	8,6016

Fonte: Dados da pesquisa

Como mencionado no início desta Seção, a aplicação EP trabalha com dados de tamanho inferior em relação às demais aplicações, prevalecendo nela o tipo de comunicação muitos-para-muitos, ou *broadcast*. Por esse motivo, como observado por Oliveira (2012), esta aplicação aumenta o fluxo *broadcast* em redes maiores com mais saltos, o que justifica a vantagem da topologia *Cluster*, cuja proposta é justamente reduzir a quantidade de saltos na rede-em-chip. Além disso, o consumo de potência foi menor quando menos núcleos são agrupados.

Sobre o fato da aplicação EP ser uma aplicação pequena, as mensagens nos pacotes de comunicação também acabam por ter tamanho menor em relação às outras aplicações,

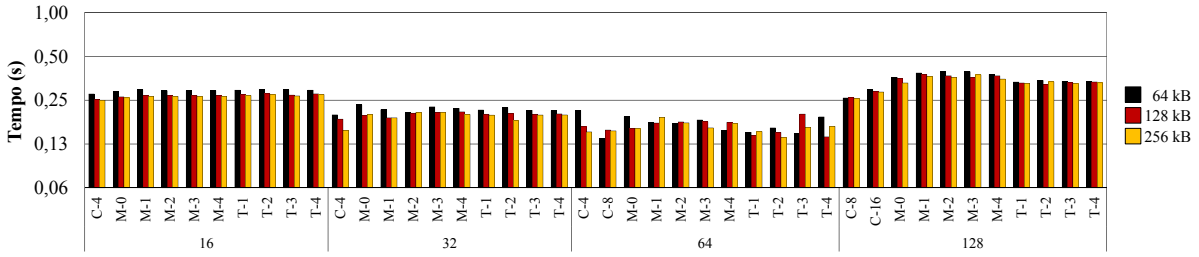
mas não o suficiente para que as *caches* de menor tamanho por núcleo tenham vantagem, considerando um fluxo de dados mais frequente.

5.3 Resultados para a aplicação FT

A aplicação FT - *Fast Fourier Transform* propõe-se a resolver numericamente equações parciais diferenciais utilizando transformadas tridimensionais de *Fourier*. Trata-se de uma aplicação que produz comunicações muitos-para-muitos (*broadcast*), com alto volume de tráfego. Para esta aplicação, tem-se os resultados para todas as quantidades de núcleos, dado que seu funcionamento não foi prejudicado em nenhuma das configurações. Dois comportamentos se repetem, em relação às aplicações CG - *Conjugate Gradient*, e EP - *Embarassingly Parallel*. O primeiro, se refere ao desempenho ruim quando utilizados 128 núcleos, em relação a quantidades menores, o que é provocado pela alta latência da rede, que é maior.

Observa-se o tempo de processamento da aplicação pelo gráfico da Figura 28 e pela Tabela 11. Para 16 núcleos, teve-se um ganho de $3,99\%$ com 64kB, na topologia C4, em relação à M0. A mesma topologia teve os melhores resultados para 128kB e 256kB, com ganhos de $3,73\%$ e $3,81\%$ respectivamente. Com 32 núcleos, permanece a topologia C4 como melhor opção em relação à M0, para a aplicação FT. Uma redução de $14,81\%$ no tempo foi obtida, com 64kB de *cache* por núcleo, nessa mesma comparação, assim como para 128kB ($5,97\%$) e para 256kB ($21,98\%$). Quando usou-se 64 núcleos e 64kB de *cache* por núcleo, a topologia que apresentou melhores resultados, em relação à M0, foi a C8, com redução no tempo de $29,08\%$. Para essa mesma quantidade, porém com 128kB, o ganho foi de $12,08\%$ com a topologia T4. E a topologia T2 apresentou $12,91\%$ de ganho em relação à topologia M0, quando o tamanho da *cache* por núcleo foi de 256kB.

Figura 28 – Aplicação FT - Tempo de processamento da aplicação



Núcleos de processamento

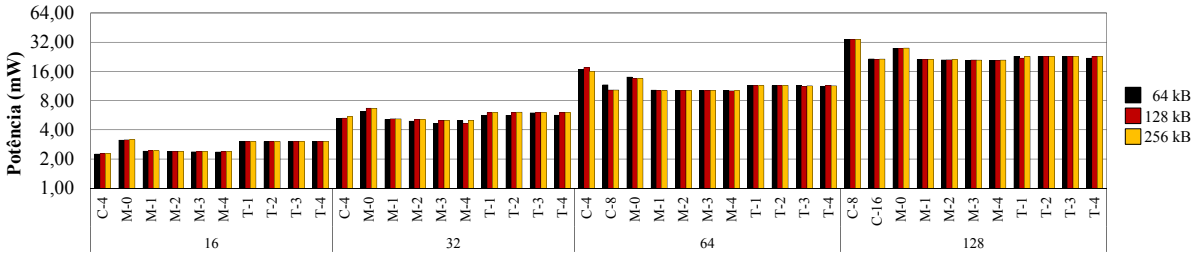
Fonte: Dados da pesquisa

Tabela 11 – Aplicação FT - Tempo de processamento da aplicação (s)

Topologia	*Cache* L2 por núcleo	Núcleos			
		16	32	64	128
C4	64 kB	0,2755	0,1981	0,2128	-
	128 kB	0,2534	0,1852	0,1640	-
	256 kB	0,2497	0,1556	0,1513	-
C8	64 kB	-	-	0,1366	0,2589
	128 kB	-	-	0,1558	0,2602
	256 kB	-	-	0,1537	0,2573
C16	64 kB	-	-	-	0,2945
	128 kB	-	-	-	0,2858
	256 kB	-	-	-	0,2843
M0	64 kB	0,2870	0,2325	0,1926	0,3590
	128 kB	0,2633	0,1969	0,1592	0,3526
	256 kB	0,2595	0,1994	0,1597	0,3281
M1	64 kB	0,2953	0,2148	0,1748	0,3841
	128 kB	0,2688	0,1889	0,1741	0,3749
	256 kB	0,2655	0,1890	0,1908	0,3651
M2	64 kB	0,2900	0,2050	0,1722	0,3941
	128 kB	0,2693	0,2036	0,1772	0,3673
	256 kB	0,2657	0,2064	0,1746	0,3592
M3	64 kB	0,2909	0,2248	0,1828	0,3918
	128 kB	0,2694	0,2067	0,1775	0,3577
	256 kB	0,2655	0,2062	0,1612	0,3734
M4	64 kB	0,2908	0,2197	0,1541	0,3762
	128 kB	0,2694	0,2077	0,1759	0,3674
	256 kB	0,2656	0,1994	0,1726	0,3491
T1	64 kB	0,2928	0,2141	0,1503	0,3332
	128 kB	0,2724	0,2004	0,1432	0,3280
	256 kB	0,2690	0,1974	0,1521	0,3263
T2	64 kB	0,2966	0,2218	0,1613	0,3411
	128 kB	0,2763	0,2021	0,1501	0,3204
	256 kB	0,2731	0,1804	0,1390	0,3344
T3	64 kB	0,2953	0,2131	0,1473	0,3362
	128 kB	0,2711	0,2004	0,2005	0,3299
	256 kB	0,2674	0,1977	0,1630	0,3251
T4	64 kB	0,2930	0,2117	0,1918	0,3348
	128 kB	0,2760	0,2012	0,1400	0,3332
	256 kB	0,2732	0,1980	0,1652	0,3290
	Melhor resultado	0,2497	0,1556	0,1366	0,2573
	Pior resultado	0,2966	0,2325	0,2128	0,3941

O segundo fator observado se refere à potência consumida, ilustrada no gráfico da Figura 29, cujos valores estão descritos na Tabela 12. A potência é novamente maior nas configurações com maior número de componentes (links e roteadores). Com 16 núcleos, os melhores resultados foram para a topologia C4, apresentando até $28,10\%$ de melhoria quando comparada à topologia M0. Com 32 núcleos, a topologia M3 se destacou em relação à M0, com redução no consumo de potência de $25,04\%$ e $24,91\%$ para 64kB e 256kB, nesta ordem. Com 128kB, o melhor resultado foi a redução de $30,65\%$ no consumo, saindo da topologia M0 para M4. A M4 apresentou boas reduções com 64 núcleos, sendo $25,47\%$ e $25,32\%$ para, respectivamente, as configurações com 128kB e 256kB de *cache* por núcleo, também em relação à M0. Com *caches* de 64kB, o ganho mais significativo foi para a topologia M3 ($27,51\%$ menos consumo do que com a M0). Por fim, com 128 núcleos, os ganhos também ficaram divididos entre as topologias M3 e M4. Para a primeira, os melhores resultados nas configurações com 64kB e 256kB ($25,09\%$ e $24,80\%$ de redução no consumo em relação à M0). Já para a segunda, ganhos de $24,46\%$ com *caches* de 128kB, também com referência à topologia M0.

Figura 29 – Aplicação FT - Potência consumida pela rede



Núcleos de processamento

Fonte: Dados da pesquisa

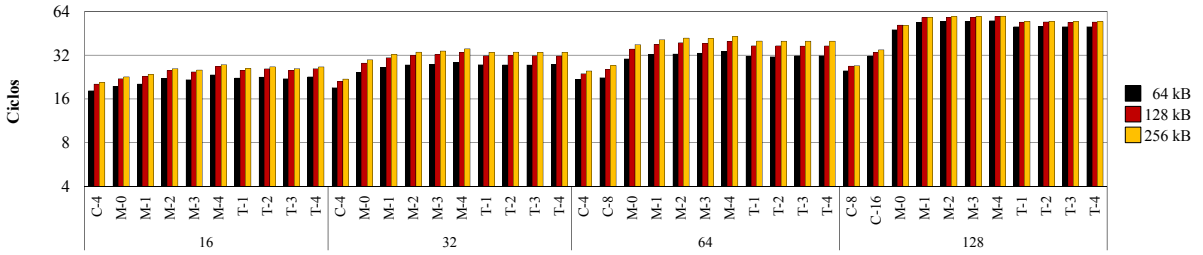
Tabela 12 – Aplicação FT - Potência consumida pela rede (mW)

Topologia	*Cache* L2 por núcleo	Núcleos			
		16	32	64	128
C4	64 kB	2,2535	5,2487	16,6701	-
	128 kB	2,2663	5,2746	17,4407	-
	256 kB	2,2675	5,4585	16,0569	-
C8	64 kB	-	-	11,6373	33,8685
	128 kB	-	-	10,2774	34,2521
	256 kB	-	-	10,2733	34,0657
C16	64 kB	-	-	-	21,5015
	128 kB	-	-	-	21,3516
	256 kB	-	-	-	21,4346
M0	64 kB	3,1315	6,1903	13,9748	27,7098
	128 kB	3,1511	6,6817	13,5173	27,5890
	256 kB	3,1537	6,6641	13,5068	27,7946
M1	64 kB	2,4136	5,1100	10,2595	21,1320
	128 kB	2,4296	5,1856	10,2303	21,2631
	256 kB	2,4305	5,1784	10,1471	21,2992
M2	64 kB	2,3926	4,9136	10,1828	20,9847
	128 kB	2,4049	5,1015	10,1495	21,0234
	256 kB	2,4072	5,0841	10,1546	21,1481
M3	64 kB	2,3741	4,6402	10,1304	20,7569
	128 kB	2,3862	5,0075	10,1266	20,9517
	256 kB	2,3881	5,0039	10,2000	20,9006
M4	64 kB	2,3655	4,9682	10,2195	20,8267
	128 kB	2,3813	4,6339	10,0746	20,8411
	256 kB	2,3836	5,0146	10,0862	20,9026
T1	64 kB	3,0163	5,6522	11,4512	22,8583
	128 kB	3,0254	6,0124	11,4665	21,9210
	256 kB	3,0272	6,0160	11,4014	22,8300
T2	64 kB	3,0251	5,6419	11,3925	22,7840
	128 kB	3,0344	6,0061	11,4277	22,8553
	256 kB	3,0357	6,0838	11,4805	22,8232
T3	64 kB	3,0128	5,9827	11,4664	22,6734
	128 kB	3,0299	6,0053	11,2040	22,7730
	256 kB	3,0314	6,0083	11,3487	22,8340
T4	64 kB	3,0403	5,6601	11,2694	21,9188
	128 kB	3,0446	6,0114	11,4911	22,7560
	256 kB	3,0450	6,0151	11,3450	22,8085
	Melhor resultado	2,2535	4,6339	10,0746	20,7569
	Pior resultado	3,1537	6,6817	17,4407	34,2521

Fonte: Dados da pesquisa

Em relação à latência média da rede, apresentada no gráfico da Figura 30 e na Tabela 13, a aplicação apresenta resultados altos, quando comparados aos da aplicação EP - *Embarassingly Parallel*. Isso está atrelado à característica de comunicações em volumes intensos da aplicação FT. Reduzir o número de saltos é uma boa estratégia nesse caso, o que justificaria um estudo de mapeamento de processos adequado. Como exemplo, as topologias do tipo *Cluster*, que apresentam melhores resultados de latência. As distâncias entre núcleos são menores, dados os agrupamentos, logo, o desempenho é melhor.

Figura 30 – Aplicação FT - Latência de rede



Núcleos de processamento

Fonte: Dados da pesquisa

Tabela 13 – Aplicação FT - Latência de rede (ciclos)

Topologia	*Cache* L2 por núcleo	Núcleos			
		16	32	64	128
C4	64 kB	18,22	19,14	21,77	-
	128 kB	20,22	21,09	23,83	-
	256 kB	20,74	21,90	24,88	-
C8	64 kB	-	-	22,39	24,96
	128 kB	-	-	25,50	26,88
	256 kB	-	-	27,16	27,06
C16	64 kB	-	-	-	31,56
	128 kB	-	-	-	33,49
	256 kB	-	-	-	34,78
M0	64 kB	19,53	24,30	30,14	47,75
	128 kB	22,03	28,17	35,18	51,42
	256 kB	22,67	29,71	37,79	51,21
M1	64 kB	20,26	26,40	32,30	53,65
	128 kB	22,92	30,68	37,87	57,92
	256 kB	23,56	32,35	40,74	58,23
M2	64 kB	22,17	27,40	32,53	54,34
	128 kB	25,08	31,82	38,87	58,21
	256 kB	25,82	33,53	41,89	58,93
M3	64 kB	21,66	27,61	33,00	54,64
	128 kB	24,51	32,27	38,51	58,20
	256 kB	25,25	34,14	41,55	58,96

Continua

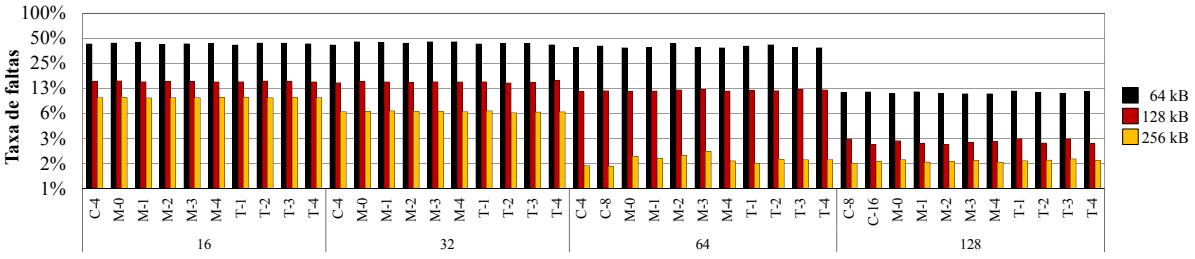
Conclusão

Topologia	*Cache* L2 por núcleo	Núcleos			
		16	32	64	128
M4	64 kB	23,38	28,46	33,96	55,06
	128 kB	26,69	33,39	39,74	58,90
	256 kB	27,50	35,31	42,91	59,26
T1	64 kB	22,27	27,47	31,48	50,14
	128 kB	25,14	31,70	37,02	53,92
	256 kB	25,88	33,46	39,96	54,65
T2	64 kB	22,59	27,40	31,24	50,28
	128 kB	25,75	31,77	37,03	54,06
	256 kB	26,58	33,54	39,90	54,69
T3	64 kB	22,07	27,37	31,68	50,22
	128 kB	25,10	31,72	36,92	53,76
	256 kB	25,85	33,49	39,91	54,38
T4	64 kB	22,68	27,56	31,73	50,14
	128 kB	25,80	31,55	36,99	53,99
	256 kB	26,55	33,48	39,90	54,51
	Melhor resultado	18,22	19,14	21,77	24,96
	Pior resultado	27,50	35,31	42,91	59,26

Fonte: Dados da pesquisa

O gráfico da Figura 31 e a Tabela 14 apresentam dados de taxa de faltas na cache L2. É mais visível, nesta aplicação, que o aumento do tamanho da memória *cache* L2 reduz a quantidade de faltas. Há também uma taxa de faltas muito elevada em relação às demais quando o tamanho da *cache* por núcleo é de 64kB até 64 núcleos. A taxa de faltas foi reduzida em valores consideráveis quando aumentou-se o tamanho por núcleo de 64kB para 128kB e 256kB, ou quando a quantidade de unidades de *cache* foi maior. Em ambos os casos, a quantidade total de memória *cache* aumentou.

Figura 31 – Aplicação FT - Taxa de faltas na L2



Núcleos de processamento

Fonte: Dados da pesquisa

Ainda que o tamanho total seja o mesmo, o tamanho das *caches* favoreceu a alocação de dados, mas isso não foi suficiente, já que o compartilhamento dessa *cache* de tamanho maior é feito por mais núcleos do que na topologia M0, por exemplo, onde cada núcleo tem uma *cache* próxima (no mesmo roteador). Para entender melhor esse comportamento, seria necessária uma avaliação específica de ocupação das *caches* (suas linhas e blocos) quando executado o programa, entendendo o funcionamento do protocolo de coerência, o que poderá ser feito em trabalho futuro.

Tabela 14 – Aplicação FT - Taxa de faltas na *cache* L2

Topologia	***Cache* L2 por núcleo**	**Núcleos**			
		16	32	64	128
C4	64 kB	42,77%	41,53%	38,81%	-
	128 kB	15,06%	14,64%	11,46%	-
	256 kB	9,65%	6,53%	1,47%	-
C8	64 kB	-	-	40,35%	11,20%
	128 kB	-	-	11,67%	3,07%
	256 kB	-	-	1,44%	1,57%
C16	64 kB	-	-	-	11,31%
	128 kB	-	-	-	2,66%
	256 kB	-	-	-	1,65%
M0	64 kB	43,77%	45,37%	38,36%	10,91%
	128 kB	15,33%	15,04%	11,51%	2,92%
	256 kB	9,71%	6,63%	1,90%	1,73%
M1	64 kB	44,31%	44,50%	38,86%	11,36%
	128 kB	14,99%	14,93%	11,61%	2,71%
	256 kB	9,58%	6,69%	1,81%	1,62%
M2	64 kB	42,12%	43,65%	43,11%	10,82%
	128 kB	15,04%	14,79%	11,97%	2,63%
	256 kB	9,66%	6,61%	1,95%	1,65%
M3	64 kB	42,91%	45,40%	38,66%	10,77%
	128 kB	15,23%	14,97%	12,12%	2,80%
	256 kB	9,65%	6,56%	2,19%	1,71%
M4	64 kB	43,26%	45,44%	38,34%	10,75%
	128 kB	14,92%	14,91%	11,59%	2,89%
	256 kB	9,73%	6,52%	1,67%	1,62%
T1	64 kB	41,44%	42,73%	40,30%	11,61%
	128 kB	15,01%	14,94%	11,82%	3,09%
	256 kB	9,68%	6,70%	1,56%	1,68%
T2	64 kB	43,73%	43,36%	41,92%	11,23%
	128 kB	15,27%	14,50%	11,73%	2,75%
	256 kB	9,59%	6,34%	1,75%	1,71%

Continua

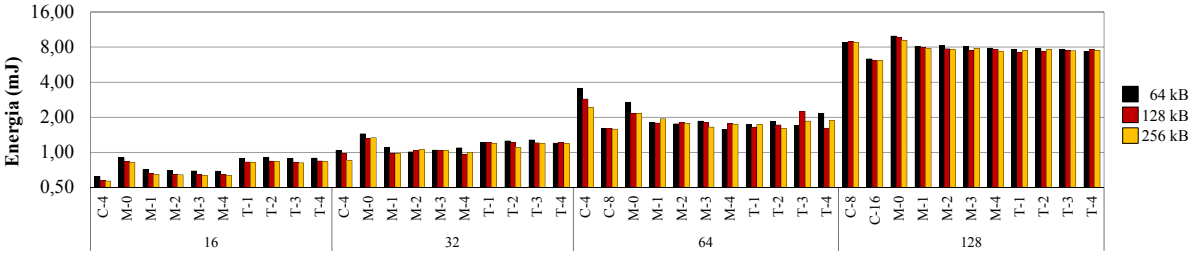
Conclusão

Topologia	*Cache* L2 por núcleo	Núcleos			
		16	32	64	128
T3	64 kB	43,65%	43,32%	39,02%	10,93%
	128 kB	15,17%	14,71%	12,15%	3,08%
	256 kB	9,71%	6,47%	1,73%	1,77%
T4	64 kB	42,85%	41,71%	38,53%	11,42%
	128 kB	14,92%	15,58%	11,95%	2,71%
	256 kB	9,67%	6,51%	1,74%	1,70%
	Melhor resultado	9,58%	6,34%	1,44%	1,57%
	Pior resultado	44,31%	45,44%	43,11%	11,61%

Fonte: Dados da pesquisa

Pela Tabela 15 e pelo gráfico na Figura 32, temos os resultados de consumo de energia para a aplicação FT. Da mesma forma que nas demais aplicações, quanto mais agrupamentos, melhor o consumo de energia. Independente da quantidade de núcleos, os melhores resultados foram obtidos com tamanho de *cache* por núcleo igual a 256kB. A aplicação FT gera comunicações com mensagens de aproximadamente 65kB (AVELAR, 2014), e, dessa forma, o maior tamanho de memória *cache* permite que essas mensagens sejam transmitidas de uma única vez, reduzindo o número de vezes que a rede-em-chip é acionada. Com 16 núcleos, o melhor resultado foi de $0,5661mJ$ de consumo de energia, com a topologia C4, assim como para 32 núcleos, que apresentou $0,8491mJ$ de consumo para essa topologia. Conforme já mencionado, aumentar a quantidade de agrupamentos provoca redução no consumo, logo, com 64 núcleos, a topologia C8 foi a melhor, com consumo de energia de $1,5794mJ$. Para 128 núcleos, a configuração com a topologia C16 consumiu $6,0936mJ$ de energia. Da mesma forma que na aplicação EP, a aplicação FT realiza comunicações do tipo *broadcast*, que acabam sendo favorecidas com a redução do número de saltos (a latência média da rede é menor, quando usadas as topologias do tipo *Cluster*).

Figura 32 – Aplicação FT - Energia consumida pela rede



Núcleos de processamento

Fonte: Dados da pesquisa

Tabela 15 – Aplicação FT - Energia consumida pela rede (mJ)

Topologia	*Cache* L2 por núcleo	Núcleos			
		16	32	64	128
C4	64 kB	0,6209	1,0395	3,5468	-
	128 kB	0,5744	0,9768	2,8605	-
	256 kB	0,5661	0,8491	2,4298	-
C8	64 kB	-	-	1,5896	8,7678
	128 kB	-	-	1,6014	8,9133
	256 kB	-	-	1,5794	8,7645
C16	64 kB	-	-	-	6,3319
	128 kB	-	-	-	6,1031
	256 kB	-	-	-	6,0936
M0	64 kB	0,8987	1,4391	2,6918	9,9482
	128 kB	0,8295	1,3158	2,1522	9,7273
	256 kB	0,8185	1,3287	2,1564	9,1181
M1	64 kB	0,7127	1,0977	1,7930	8,1162
	128 kB	0,6531	0,9797	1,7810	7,9715
	256 kB	0,6454	0,9787	1,9359	7,7766
M2	64 kB	0,6938	1,0072	1,7536	8,2697
	128 kB	0,6477	1,0387	1,7989	7,7212
	256 kB	0,6396	1,0495	1,7725	7,5967
M3	64 kB	0,6907	1,0432	1,8515	8,1318
	128 kB	0,6428	1,0351	1,7978	7,4945
	256 kB	0,6340	1,0319	1,6439	7,8033
M4	64 kB	0,6879	1,0916	1,5749	7,8347
	128 kB	0,6414	0,9626	1,7720	7,6562
	256 kB	0,6330	1,0000	1,7411	7,2963
T1	64 kB	0,8833	1,2100	1,7214	7,6156
	128 kB	0,8240	1,2051	1,6420	7,1895
	256 kB	0,8142	1,1878	1,7342	7,4499
T2	64 kB	0,8974	1,2515	1,8373	7,7725
	128 kB	0,8383	1,2140	1,7153	7,3231
	256 kB	0,8289	1,0978	1,5963	7,6316
T3	64 kB	0,8897	1,2752	1,6892	7,6218
	128 kB	0,8214	1,2037	2,2460	7,5118
	256 kB	0,8105	1,1876	1,8495	7,4241
T4	64 kB	0,8908	1,1985	2,1611	7,3380
	128 kB	0,8405	1,2094	1,6085	7,5827
	256 kB	0,8320	1,1908	1,8747	7,5031
	Melhor resultado	0,5661	0,8491	1,5749	6,0936
	Pior resultado	0,8987	1,4391	3,5468	9,9482

Fonte: Dados da pesquisa

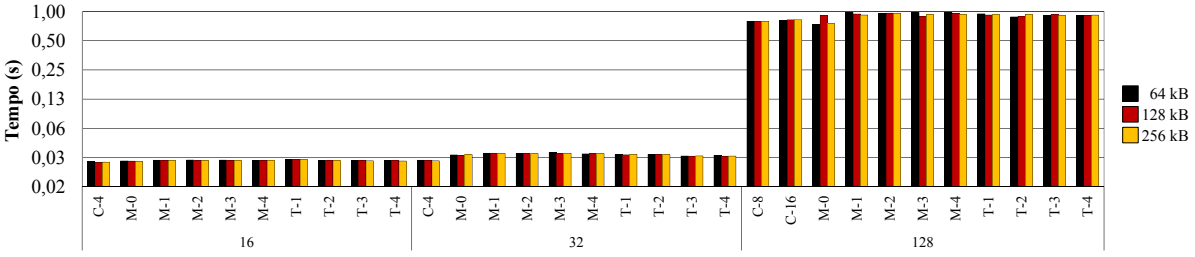
5.4 Resultados para a aplicação IS

A aplicação IS - *Integer Sort* propõe-se a gerar e ordenar números inteiros. É considerada uma aplicação com tamanho de dados intermediário, quando comparada às demais aplicações. Além disso, realiza acessos aleatórios à memória, com característica irregular – as computações paralelas não possuem a mesma intensidade de trabalho. Os padrões de comunicação dessa aplicação são mistos, mas prevalecem as comunicações muitos-pra-um, o que é resultado da estratégia de ordenação *bucket sort*, como será destacado ao final desta Seção.

Pelo gráfico da Figura 33, e pela Tabela 16, é possível, novamente, observar nesta aplicação a desvantagem em executar o algoritmo com uma grande quantidade de núcleos. O tempo gasto com comunicações ultrapassa aquele gasto com computações. Conforme pode ser observado no gráfico da Figura 34 e na Tabela 17, há um aumento da latência da rede e da concorrência pelos recursos à medida em que a quantidade de núcleos também aumenta. Mais uma vez destaca-se a importância do estudo de estratégias como mapeamento de processos, alinhando a análise e alteração de algoritmos e da arquitetura, de maneira conjunta.

Com aplicação IS foram observados dois casos em que não houve ganhos em relação à topologia M0, sendo ela a melhor opção. Os casos referem-se ao uso de 128 núcleos, com 64kB e 256kB de *cache* por núcleo. Já para 128kB, houve uma redução de $13,98\%$ na topologia C8, em relação à M0. Reduzindo a quantidade de núcleos para 32 ou 16, os melhores resultados de tempo foram encontrados na topologia C4. Com 32 núcleos, houve redução de $1,82\%$ até $2,79\%$ quando comparada essa topologia C4 em relação à M0. Já para 32 núcleos, as reduções no tempo de processamento da aplicação foram de $11,82\%$, $11,53\%$ e $13,38\%$, para 64kB, 128kB e 256kB, respectivamente, também relacionado à topologia M0.

Figura 33 – Aplicação IS - Tempo de processamento da aplicação



Núcleos de processamento

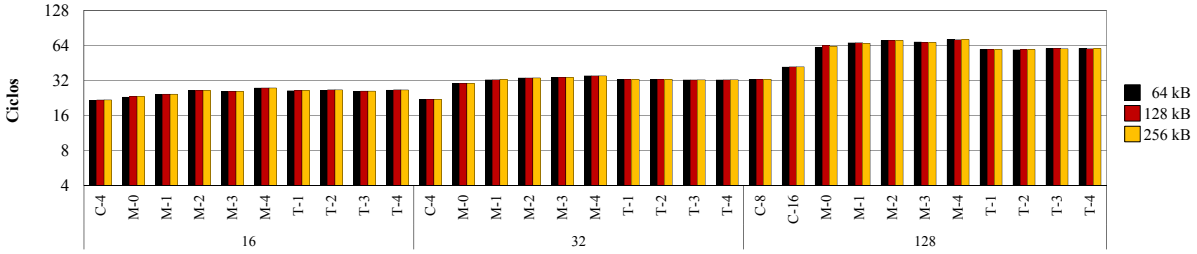
Fonte: Dados da pesquisa

Tabela 16 – Aplicação IS - Tempo de processamento da aplicação (s)

Topologia	*Cache* L2 por núcleo	Núcleos		
		16	32	128
C4	64 kB	0,0283	0,0294	-
	128 kB	0,0279	0,0292	-
	256 kB	0,0280	0,0289	-
C8	64 kB	-	-	0,7914
	128 kB	-	-	0,7880
	256 kB	-	-	0,7899
C16	64 kB	-	-	0,8071
	128 kB	-	-	0,8207
	256 kB	-	-	0,8203
M0	64 kB	0,0288	0,0333	0,7309
	128 kB	0,0287	0,0330	0,9160
	256 kB	0,0286	0,0334	0,7467
M1	64 kB	0,0294	0,0347	0,9725
	128 kB	0,0293	0,0347	0,9413
	256 kB	0,0293	0,0346	0,9184
M2	64 kB	0,0295	0,0346	0,9514
	128 kB	0,0291	0,0344	0,9608
	256 kB	0,0291	0,0344	0,9560
M3	64 kB	0,0294	0,0352	0,9798
	128 kB	0,0293	0,0345	0,8927
	256 kB	0,0291	0,0347	0,9303
M4	64 kB	0,0293	0,0341	0,9813
	128 kB	0,0292	0,0346	0,9531
	256 kB	0,0292	0,0344	0,9270
T1	64 kB	0,0298	0,0335	0,9449
	128 kB	0,0297	0,0334	0,9188
	256 kB	0,0299	0,0337	0,9258
T2	64 kB	0,0293	0,0336	0,8789
	128 kB	0,0290	0,0337	0,8890
	256 kB	0,0290	0,0335	0,9342
T3	64 kB	0,0290	0,0324	0,9128
	128 kB	0,0291	0,0323	0,9256
	256 kB	0,0290	0,0325	0,9112
T4	64 kB	0,0292	0,0330	0,9097
	128 kB	0,0290	0,0323	0,9121
	256 kB	0,0287	0,0324	0,9179
	Melhor resultado	0,0279	0,0289	0,7309
	Pior resultado	0,0299	0,0352	0,9813

Fonte: Dados da pesquisa

Figura 34 – Aplicação IS - Latência de rede



Núcleos de processamento

Fonte: Dados da pesquisa

Tabela 17 – Aplicação IS - Latência de rede (ciclos)

Topologia	*Cache* L2 por núcleo	Núcleos		
		16	32	128
C4	64 kB	21,54	22,03	-
	128 kB	21,78	22,09	-
	256 kB	21,83	22,08	-
C8	64 kB	-	-	32,76
	128 kB	-	-	32,73
	256 kB	-	-	32,81
C16	64 kB	-	-	41,79
	128 kB	-	-	41,95
	256 kB	-	-	41,99
M0	64 kB	23,06	30,20	61,40
	128 kB	23,26	30,28	64,00
	256 kB	23,26	30,22	62,87
M1	64 kB	24,31	32,42	67,39
	128 kB	24,45	32,50	67,44
	256 kB	24,45	32,60	66,86
M2	64 kB	26,17	33,65	70,58
	128 kB	26,31	33,64	70,99
	256 kB	26,21	33,72	70,82
M3	64 kB	25,65	33,92	68,53
	128 kB	25,79	34,02	68,20
	256 kB	25,84	34,07	67,94
M4	64 kB	27,60	35,10	72,07
	128 kB	27,64	35,11	71,57
	256 kB	27,66	35,11	71,96
T1	64 kB	26,12	32,66	59,21
	128 kB	26,27	32,70	59,08
	256 kB	26,25	32,69	59,07

Continua

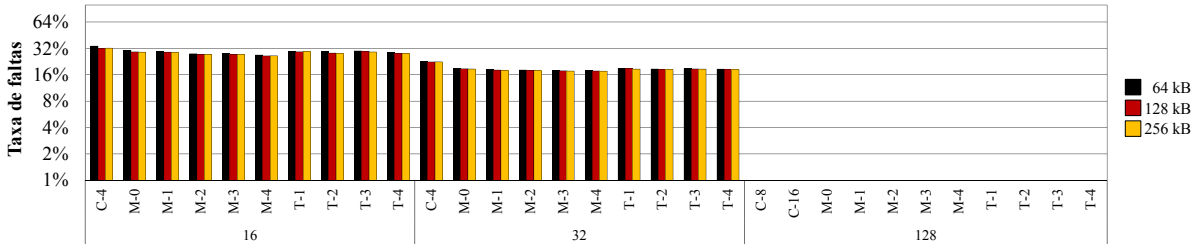
Conclusão

Topologia	*Cache* L2 por núcleo	Núcleos		
		16	32	128
T2	64 kB	26,35	32,67	58,64
	128 kB	26,61	32,77	58,85
	256 kB	26,64	32,76	58,97
T3	64 kB	25,77	32,36	60,12
	128 kB	25,96	32,39	60,10
	256 kB	25,99	32,42	59,93
T4	64 kB	26,45	32,37	60,30
	128 kB	26,61	32,44	59,96
	256 kB	26,60	32,50	60,06
	Melhor resultado	21,54	22,03	32,73
	Pior resultado	27,66	35,11	72,07

Fonte: Dados da pesquisa

Observando a taxa de faltas pelo gráfico da Figura 35 e pela Tabela 18, resultado de destaque foi notado nesta aplicação quando utilizados 128 núcleos. Ainda que o desempenho tenha sido prejudicado pela latência da rede e gasto com comunicações, a taxa de faltas reduziu muito com o aumento da quantidade de núcleos para 128. Isso se justifica pelo aumento proporcional na quantidade total de memória *cache* L2 disponível, que está associada à quantidade de núcleos (os tamanhos apresentados de 64kB, 128kB e 256kB são por núcleo). As quantidades totais de cache L2 maiores do que 8MB mostram-se suficientes para que a aplicação IS apresente bons resultados em termos de taxa de faltas (próximos de 0%), ainda que os acessos sejam aleatórios.

Figura 35 – Aplicação IS - Taxa de faltas na L2



Núcleos de processamento

Fonte: Dados da pesquisa

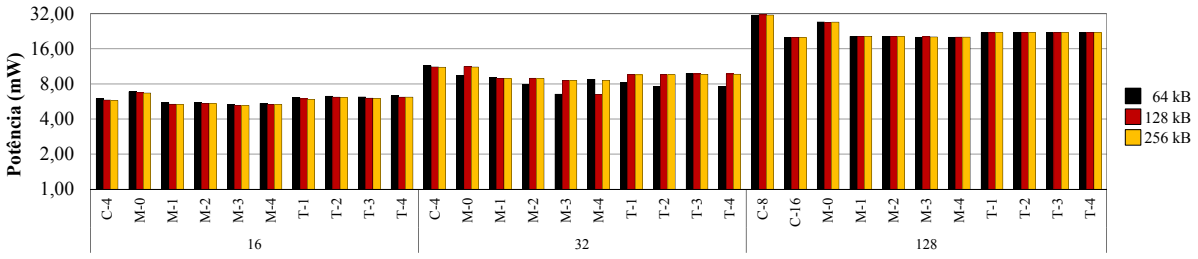
Tabela 18 – Aplicação IS - Taxa de faltas na *cache* L2

Topologia	*Cache* L2 por núcleo	Núcleos		
		16	32	128
C4	64 kB	33,95%	22,66%	-
	128 kB	32,12%	22,39%	-
	256 kB	32,19%	22,44%	-
C8	64 kB	-	-	0,39%
	128 kB	-	-	0,39%
	256 kB	-	-	0,38%
C16	64 kB	-	-	0,35%
	128 kB	-	-	0,34%
	256 kB	-	-	0,31%
M0	64 kB	30,35%	18,79%	0,28%
	128 kB	29,17%	18,72%	0,26%
	256 kB	28,97%	18,59%	0,28%
M1	64 kB	29,37%	18,38%	0,27%
	128 kB	29,00%	18,11%	0,27%
	256 kB	28,62%	17,87%	0,27%
M2	64 kB	27,75%	18,12%	0,28%
	128 kB	27,13%	17,87%	0,26%
	256 kB	27,33%	17,78%	0,26%
M3	64 kB	28,13%	18,01%	0,28%
	128 kB	27,19%	17,76%	0,27%
	256 kB	27,22%	17,63%	0,25%
M4	64 kB	26,83%	17,82%	0,27%
	128 kB	26,28%	17,66%	0,25%
	256 kB	26,31%	17,60%	0,25%
T1	64 kB	29,73%	18,83%	0,28%
	128 kB	29,24%	18,75%	0,26%
	256 kB	29,38%	18,56%	0,26%
T2	64 kB	29,71%	18,62%	0,27%
	128 kB	28,25%	18,40%	0,26%
	256 kB	27,95%	18,30%	0,26%
T3	64 kB	29,89%	18,74%	0,28%
	128 kB	29,62%	18,63%	0,28%
	256 kB	29,09%	18,55%	0,27%
T4	64 kB	28,86%	18,57%	0,28%
	128 kB	28,17%	18,39%	0,27%
	256 kB	27,98%	18,28%	0,27%
	Melhor resultado	26,28%	17,60%	0,25%
	Pior resultado	33,95%	22,66%	0,39%

Fonte: Dados da pesquisa

No gráfico da Figura 36 estão os resultados de potência consumida pela rede para a aplicação IS, assim como na Tabela 19. Quando são usados 16 núcleos, a topologia que apresentou a maior redução no consumo de potência foi a M3 (até $21,87\%$ de redução em relação à topologia M0), independente do tamanho de *cache* por núcleo. A topologia M3 também apresentou reduções no consumo em relação à topologia M0, quando usados 32 núcleos. Os valores foram reduzidos em $30,92\%$ e $23,38\%$ com *caches* de 64kB e 256kB por núcleo, respectivamente. Já para 128kB de *cache* por núcleo, a maior redução no consumo a partir da M0 foi encontrada com a topologia M4, sendo $42,60\%$. Com 128 núcleos, todos os melhores resultados foram para a topologia C16, com reduções de $25,44\%$ a $26,38\%$ em relação à topologia M0.

Figura 36 – Aplicação IS - Potência consumida pela rede



Núcleos de processamento

Fonte: Dados da pesquisa

Tabela 19 – Aplicação IS - Potência consumida pela rede (mW)

Topologia	*Cache* L2 por núcleo	Núcleos		
		16	32	128
C4	64 kB	5,9787	11,4500	-
	128 kB	5,8235	11,1964	-
	256 kB	5,7563	11,1526	-
C8	64 kB	-	-	31,1951
	128 kB	-	-	31,3392
	256 kB	-	-	31,2557
C16	64 kB	-	-	20,0960
	128 kB	-	-	19,9826
	256 kB	-	-	20,0075
M0	64 kB	6,8404	9,4695	27,2524
	128 kB	6,7029	11,2804	26,8003
	256 kB	6,6907	11,1767	27,1755
M1	64 kB	5,5061	9,0538	20,4462
	128 kB	5,3618	8,9178	20,5203
	256 kB	5,3306	8,8902	20,5388

Continua

Conclusão

Topologia	*Cache* L2 por núcleo	Núcleos		
		16	32	128
M2	64 kB	5,5197	7,8533	20,4081
	128 kB	5,4294	8,9701	20,3639
	256 kB	5,4094	8,9252	20,3985
M3	64 kB	5,3532	6,5419	20,1849
	128 kB	5,2369	8,6126	20,2995
	256 kB	5,2325	8,5637	20,2614
M4	64 kB	5,4484	8,7815	20,1712
	128 kB	5,3571	6,4755	20,1894
	256 kB	5,3327	8,6152	20,2102
T1	64 kB	6,0931	8,2279	22,0724
	128 kB	5,9503	9,6601	21,9699
	256 kB	5,9085	9,5732	22,0673
T2	64 kB	6,2842	7,6285	22,1251
	128 kB	6,1669	9,6148	22,1021
	256 kB	6,1326	9,6107	22,0713
T3	64 kB	6,1832	9,8672	22,0510
	128 kB	6,0426	9,7551	22,0627
	256 kB	6,0325	9,6958	22,0895
T4	64 kB	6,3156	7,6325	22,0045
	128 kB	6,1699	9,7568	22,0821
	256 kB	6,1720	9,7013	22,0886
	Melhor resultado	5,2325	6,4755	19,9826
	Pior resultado	6,8404	11,4500	31,3392

Fonte: Dados da pesquisa

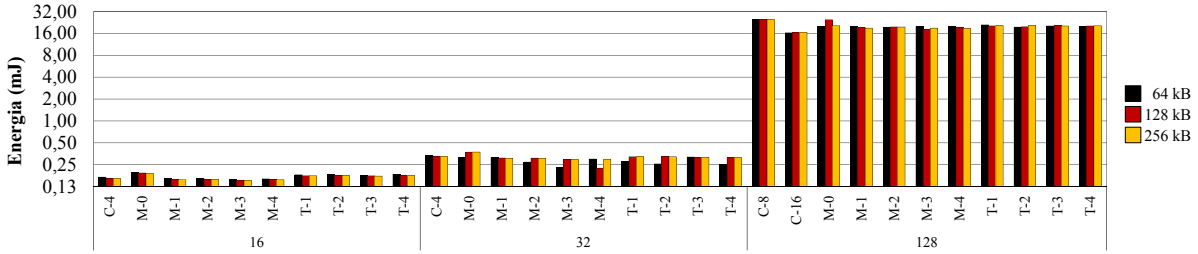
Diferente das demais aplicações, a IS não teve os melhores resultados concentrados na topologia do tipo *Cluster*. Pelo gráfico da Figura 37 e pela Tabela 20, observa-se que apenas com a maior quantidade de núcleos (128) é que uma configuração com topologia *Cluster* teve resultados melhores: a topologia C16, com 64kB de *cache* L2 por núcleo, consumiu $16,2197mJ$ de energia. Com 16 núcleos, o melhor consumo foi de $0,1520mJ$ na configuração com a topologia do tipo *Mesh* (M3) e com 256kB de *cache* L2 por núcleo. Por fim, para 32 núcleos, o melhor resultado foi obtido na configuração com a topologia M4, com 128kB de *cache* L2 por núcleo, sendo $0,2242mJ$ de consumo energético.

A aplicação IS, como observado por Oliveira (2012), é caracterizada também por comunicações muitos-para-um, isto é, vários núcleos de processamento demandam comunicações com um mesmo núcleo de destino. Essa característica é derivada da estratégia de ordenação *bucket sort*, que faz uso da estratégia *merge sort*, e é comum encontrar versões paralelas de algoritmos de ordenação que fazem esse tipo de comunicação. Dessa maneira, um determinado roteador tem que terminar um roteamento

para iniciar outro, já que várias outras comunicações podem ter sido direcionadas àquele roteador. Isso pode gerar gargalos nos roteadores e rotas, então, neste caso, ter rotas alternativas disponíveis pode favorecer a comunicação, assim como ocorre nas *Meshs* em relação às *Clusters*. Mas, ainda assim, a latência da rede é maior.

IS também é considerada uma aplicação com tamanho de dados intermediário, realizando acessos aleatórios à memória. Dessa forma, o desempenho depende do fluxo de dados transmitido em cada momento da computação, e o tamanho de *cache* ideal pode variar. De maneira geral, foram obtidos os melhores resultados de consumo energético com as topologias M3 e M4, principalmente quando a quantidade de núcleos é menor. Com mais núcleos, da mesma forma que nas outras aplicações, a latência da rede aumenta muito, e agrupar se mostra novamente uma estratégia viável, reduzindo a latência, o tempo de processamento da aplicação, e consequentemente o consumo energético.

Figura 37 – Aplicação IS - Energia consumida pela rede



Núcleos de processamento

Fonte: Dados da pesquisa

Tabela 20 – Aplicação IS - Energia consumida pela rede (mJ)

Topologia	*Cache* L2 por núcleo	Núcleos		
		16	32	128
C4	64 kB	0,1692	0,3363	-
	128 kB	0,1625	0,3269	-
	256 kB	0,1614	0,3226	-
C8	64 kB	-	-	24,6876
	128 kB	-	-	24,6940
	256 kB	-	-	24,6902
C16	64 kB	-	-	16,2197
	128 kB	-	-	16,3999
	256 kB	-	-	16,4123

Continua

Conclusão

Topologia	***Cache* L2 por núcleo**	**Núcleos**		
		16	32	128
M0	64 kB	0,1972	0,3154	19,9175
	128 kB	0,1924	0,3723	24,5502
	256 kB	0,1911	0,3732	20,2926
M1	64 kB	0,1616	0,3138	19,8831
	128 kB	0,1568	0,3091	19,3157
	256 kB	0,1559	0,3075	18,8618
M2	64 kB	0,1626	0,2718	19,4163
	128 kB	0,1580	0,3082	19,5650
	256 kB	0,1573	0,3068	19,5006
M3	64 kB	0,1574	0,2303	19,7781
	128 kB	0,1533	0,2972	18,1208
	256 kB	0,1520	0,2968	18,8496
M4	64 kB	0,1597	0,2996	19,7949
	128 kB	0,1565	0,2242	19,2422
	256 kB	0,1558	0,2963	18,7351
T1	64 kB	0,1816	0,2759	20,8557
	128 kB	0,1768	0,3222	20,1852
	256 kB	0,1766	0,3227	20,4293
T2	64 kB	0,1842	0,2566	19,4458
	128 kB	0,1788	0,3239	19,6492
	256 kB	0,1781	0,3216	20,6200
T3	64 kB	0,1791	0,3198	20,1281
	128 kB	0,1761	0,3153	20,4222
	256 kB	0,1746	0,3146	20,1272
T4	64 kB	0,1844	0,2518	20,0175
	128 kB	0,1791	0,3152	20,1419
	256 kB	0,1774	0,3141	20,2757
	Melhor resultado	0,1520	0,2242	16,2197
	Pior resultado	0,1972	0,3732	24,6940

Fonte: Dados da pesquisa

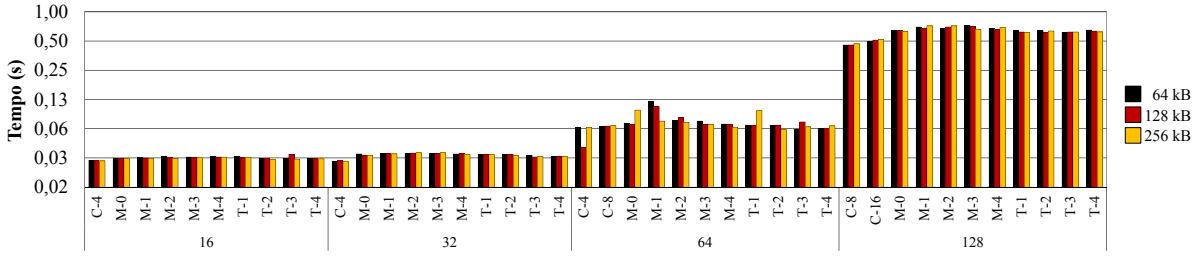
5.5 Resultados para a aplicação MG

A aplicação MG - *Multigrid* permite obter uma solução aproximada para uma equação tridimensional discreta de *Poisson*, utilizando o método multigrade Ciclo-V. É uma comunicação que faz acessos intensos à memória. O padrão de comunicação é variável de acordo com a quantidade de núcleos disponíveis – com menos de 64 núcleos na rede, prevalecem as comunicações do tipo *unicast*, ou um-para-um; com mais núcleos do que isso, prevalecem as comunicações do tipo *broadcast* (OLIVEIRA, 2012). Além disso, é uma aplicação caracterizada pelas comunicações de longa e pequena distância, isto é, não produzi comunicações entre núcleos com distâncias intermediárias.

Esta aplicação também não apresentou escalabilidade quando a quantidade de núcleos foi elevada para 128, como pode-se notar no gráfico da Figura 38 e pela Tabela 21. Para as configurações com menores quantidades de núcleos, a topologia C4 se destaca em relação às demais. Foram percebidas reduções de até $5,40\%$ quando foram usados 16 núcleos nesta topologia, em relação à topologia M0. Já com 32 núcleos, alcançou-se uma redução de até $15,59\%$ no tempo de processamento da aplicação, também com a topologia C4.

Quando optou-se por usar 64 núcleos, a topologia C4 também proporcionou a maior redução no tempo em relação à M0, com $41,75\%$ de redução para a configuração com 128kB de memória *cache* L2 por núcleo. Com 64kB, o melhor resultado foi da topologia T3, sendo $13,02\%$ de redução. E com 256kB, houve $37,12\%$ de redução com a topologia T2. Com 128 núcleos, os melhores resultados foram todos obtidos quando se usou a topologia C8, sendo $29,02\%$, $28,71\%$ e $25,42\%$ de redução para, respectivamente, 64kB, 128kB e 256kB de *cache* L2 por núcleo, com referência à topologia M0.

Figura 38 – Aplicação MG - Tempo de processamento da aplicação



Núcleos de processamento

Fonte: Dados da pesquisa

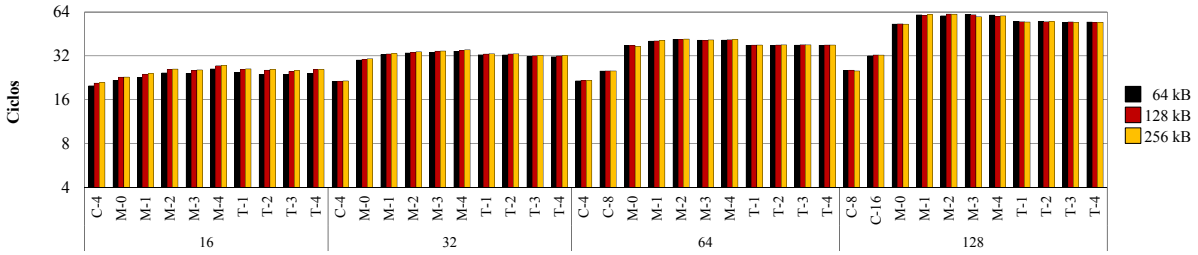
Tabela 21 – Aplicação MG - Tempo de processamento da aplicação (s)

Topologia	*Cache* L2 por núcleo	Núcleos			
		16	32	64	128
C4	64 kB	0,0295	0,0289	0,0645	-
	128 kB	0,0291	0,0291	0,0401	-
	256 kB	0,0291	0,0290	0,0645	-
C8	64 kB	-	-	0,0662	0,4547
	128 kB	-	-	0,0664	0,4556
	256 kB	-	-	0,0670	0,4676
C16	64 kB	-	-	-	0,4979
	128 kB	-	-	-	0,5111
	256 kB	-	-	-	0,5196
M0	64 kB	0,0311	0,0343	0,0710	0,6405
	128 kB	0,0307	0,0333	0,0689	0,6391
	256 kB	0,0307	0,0334	0,0972	0,6270
M1	64 kB	0,0318	0,0344	0,1187	0,6957
	128 kB	0,0312	0,0349	0,1056	0,6740
	256 kB	0,0309	0,0344	0,0748	0,7155
M2	64 kB	0,0320	0,0346	0,0764	0,6729
	128 kB	0,0318	0,0350	0,0808	0,6981
	256 kB	0,0309	0,0353	0,0729	0,7161
M3	64 kB	0,0319	0,0348	0,0749	0,7179
	128 kB	0,0315	0,0348	0,0689	0,7030
	256 kB	0,0313	0,0353	0,0691	0,6601
M4	64 kB	0,0324	0,0343	0,0698	0,6787
	128 kB	0,0319	0,0343	0,0693	0,6521
	256 kB	0,0318	0,0339	0,0649	0,6887
T1	64 kB	0,0320	0,0340	0,0669	0,6423
	128 kB	0,0315	0,0336	0,0670	0,6190
	256 kB	0,0315	0,0336	0,0957	0,6163
T2	64 kB	0,0311	0,0336	0,0679	0,6395
	128 kB	0,0306	0,0340	0,0680	0,6171
	256 kB	0,0301	0,0331	0,0611	0,6341
T3	64 kB	0,0309	0,0331	0,0618	0,6106
	128 kB	0,0341	0,0320	0,0733	0,6194
	256 kB	0,0304	0,0322	0,0652	0,6188
T4	64 kB	0,0308	0,0322	0,0629	0,6485
	128 kB	0,0306	0,0322	0,0621	0,6331
	256 kB	0,0306	0,0323	0,0668	0,6207
	Melhor resultado	0,0291	0,0289	0,0401	0,4547
	Pior resultado	0,0341	0,0353	0,1187	0,7179

Fonte: Dados da pesquisa

A latência média da rede, ilustrada no gráfico da Figura 39, e descrita na Tabela 22, tem melhores resultados nas abordagens que agrupam núcleos (*Cluster*), justamente por centralizar os acessos à memória, que, na aplicação MG, são feitos de forma intensa.

Figura 39 – Aplicação MG - Latência de rede



Núcleos de processamento

Fonte: Dados da pesquisa

Tabela 22 – Aplicação MG - Latência de rede (ciclos)

Topologia	***Cache* L2 por núcleo**	**Núcleos**			
		16	32	64	128
C4	64 kB	19,94	21,25	21,55	-
	128 kB	20,80	21,44	21,75	-
	256 kB	20,95	21,52	21,65	-
C8	64 kB	-	-	25,17	25,36
	128 kB	-	-	25,21	25,52
	256 kB	-	-	25,20	25,21
C16	64 kB	-	-	-	32,00
	128 kB	-	-	-	32,43
	256 kB	-	-	-	32,41
M0	64 kB	21,74	29,94	37,65	52,88
	128 kB	22,84	30,20	37,67	53,01
	256 kB	22,88	30,55	37,17	52,86
M1	64 kB	22,79	32,63	40,39	61,17
	128 kB	23,95	32,98	40,51	60,42
	256 kB	24,14	33,13	40,85	61,36

Continua

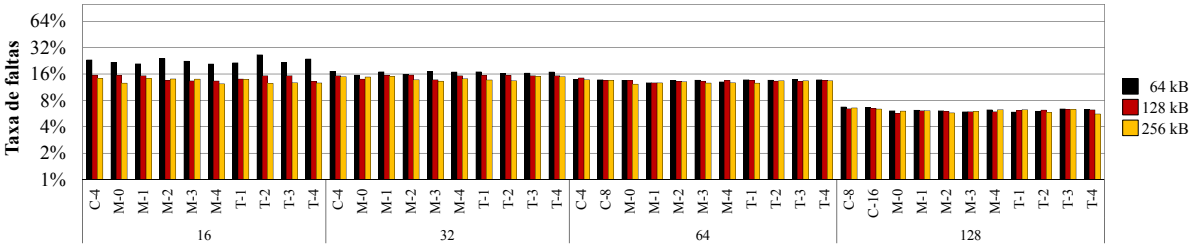
Conclusão

Topologia	*Cache* L2 por núcleo	Núcleos			
		16	32	64	128
M2	64 kB	24,44	33,50	41,54	60,37
	128 kB	25,78	33,76	41,50	61,93
	256 kB	25,98	34,12	41,75	61,89
M3	64 kB	24,24	33,85	40,99	61,55
	128 kB	25,46	34,48	40,98	61,25
	256 kB	25,62	34,55	41,15	59,39
M4	64 kB	26,03	34,43	41,04	61,04
	128 kB	27,36	34,75	41,27	59,56
	256 kB	27,52	35,22	41,45	60,18
T1	64 kB	24,68	32,56	37,88	55,09
	128 kB	25,79	32,93	37,92	54,86
	256 kB	26,02	33,07	37,95	54,67
T2	64 kB	23,90	32,47	37,80	55,15
	128 kB	25,51	32,94	37,95	54,83
	256 kB	25,87	33,02	38,05	54,92
T3	64 kB	23,88	31,66	37,91	54,23
	128 kB	24,96	31,99	38,12	54,69
	256 kB	25,32	32,20	38,06	54,26
T4	64 kB	24,15	31,55	37,87	54,61
	128 kB	25,72	31,94	37,98	54,32
	256 kB	25,81	32,15	37,99	54,46
	Melhor resultado	19,94	21,25	21,55	25,21
	Pior resultado	27,52	35,22	41,75	61,93

Fonte: Dados da pesquisa

No que se refere à taxa de faltas, apresentada no gráfico da Figura 40 e na Tabela 23, a aplicação MG apresentou taxas irregulares mas equilibradas nas configurações (com poucas alterações).

Figura 40 – Aplicação MG - Taxa de faltas na L2



Núcleos de processamento

Fonte: Dados da pesquisa

Nota-se, entretanto, que nas configurações com 128 núcleos, nas quais a quantidade total de memória cache L2 é maior, a taxa de faltas foi menor que nas demais configurações. Outra exceção são as configurações com 16 núcleos e 64kB de cache L2 por núcleo, com um tamanho reduzido para essa aplicação, apresentando taxas mais altas. A aplicação MG é caracterizada por acessos intensos à memória, o que justifica o aumento na latência da rede quando mudou-se da configuração com a topologia *Mesh* inicial (M0) para as demais topologias *Mesh*. O uso da rede para acesso à memória foi intensificado. Centralizar o acesso (topologia *Cluster*) ou disponibilizar mais caminhos (*Torus*) pode favorecer à comunicação com as memórias *cache* L2 que foram distribuídas.

Tabela 23 – Aplicação MG - Taxa de faltas na *cache* L2

Topologia	***Cache* L2 por núcleo**	**Núcleos**			
		16	32	64	128
C4	64 kB	23,19%	17,21%	13,96%	-
	128 kB	15,55%	15,30%	14,42%	-
	256 kB	14,22%	14,91%	13,78%	-
C8	64 kB	-	-	13,78%	6,75%
	128 kB	-	-	13,60%	6,45%
	256 kB	-	-	13,60%	6,56%
C16	64 kB	-	-	-	6,66%
	128 kB	-	-	-	6,52%
	256 kB	-	-	-	6,38%
M0	64 kB	21,79%	15,53%	13,60%	6,11%
	128 kB	15,49%	14,01%	13,52%	5,68%
	256 kB	12,60%	14,79%	12,27%	6,04%
M1	64 kB	20,93%	16,94%	12,76%	6,19%
	128 kB	15,29%	15,46%	12,67%	6,12%
	256 kB	14,21%	14,97%	12,66%	6,09%
M2	64 kB	24,03%	15,87%	13,48%	6,11%
	128 kB	13,54%	15,57%	13,25%	6,02%
	256 kB	14,07%	13,74%	13,10%	5,76%
M3	64 kB	22,46%	17,17%	13,62%	5,95%
	128 kB	13,33%	13,73%	13,27%	5,96%
	256 kB	13,99%	13,26%	12,63%	5,99%
M4	64 kB	20,86%	16,88%	13,02%	6,24%
	128 kB	13,33%	15,33%	13,58%	5,93%
	256 kB	12,37%	14,11%	12,70%	6,28%
T1	64 kB	21,50%	16,95%	13,82%	5,89%
	128 kB	14,02%	15,48%	13,48%	6,13%
	256 kB	13,96%	13,67%	12,60%	6,29%

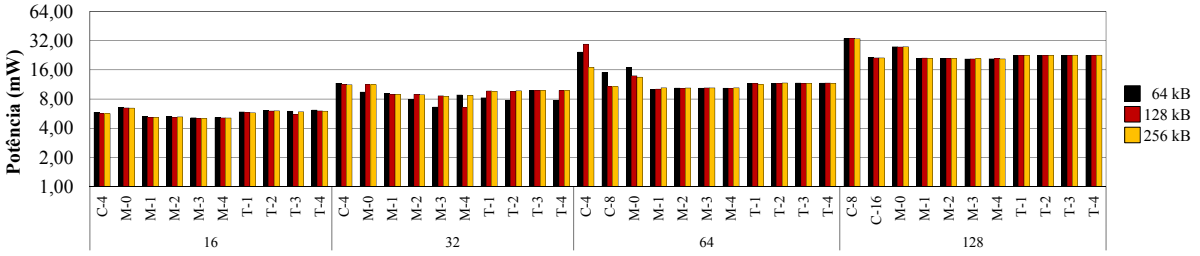
Continua

Conclusão

Topologia	*Cache* L2 por núcleo	Núcleos			
		16	32	64	128
T2	64 kB	26,64%	16,30%	13,62%	6,03%
	128 kB	15,31%	15,49%	13,28%	6,21%
	256 kB	12,53%	13,39%	13,42%	5,82%
T3	64 kB	21,79%	16,40%	13,92%	6,41%
	128 kB	15,36%	15,32%	13,22%	6,35%
	256 kB	12,72%	15,03%	13,42%	6,33%
T4	64 kB	23,78%	16,89%	13,80%	6,33%
	128 kB	13,22%	15,31%	13,54%	6,24%
	256 kB	12,66%	14,94%	13,40%	5,60%
	Melhor resultado	12,37%	13,26%	12,27%	5,60%
	Pior resultado	26,64%	17,21%	14,42%	6,75%

Fonte: Dados da pesquisa

Figura 41 – Aplicação MG - Potência consumida pela rede



Núcleos de processamento

Fonte: Dados da pesquisa

Os dados de consumo de potência da rede-em-chip são apresentados no gráfico da Figura 41 e na Tabela 24. Novamente, nota-se o consumo elevado para as topologias do tipo M0, com mais componentes de rede (links e roteadores). Entretanto, as demais topologias do tipo *Mesh* apresentaram os melhores resultados de consumo energético para esta aplicação. A topologia M1 apresentou redução de $40,45\%$ e $27,09\%$ quando usou-se 64 núcleos e, respectivamente, tamanhos de 64kB e 128kB de *cache* por núcleo, em relação à topologia M0. Com essa mesma quantidade de núcleos, porém 256kB de *cache*, o melhor resultado foi para a topologia M2 ($22,59\%$ de redução no consumo em relação à M0). Com 16 núcleos, a topologia M3 proporcionou redução no consumo de potência relativo à M0 de $22,16\%$, com 64kB, e de $22,10\%$ com 128kB de *cache* por núcleo. Com 256kB, o melhor resultado foi de $21,93\%$ de redução quando alterada a configuração de M0 para M4. Com 32 núcleos, M3 foi melhor do que a M0, para 64kB e 256kB, com reduções de $29,88\%$ e $24,10\%$, nesta ordem. Com 128kB por núcleo, o resultado mais significativo é para a topologia M4, com 42,09% de redução no consumo, também em relação à M0. Na maior quantidade de núcleos, 128, novamente os resultados se dividem entre as topologias

M3 e M4. A primeira com 25, 17% na configuração com 64kB, e 24, 72% com 128kB, enquanto a segunda foi a melhor com 25, 06% de redução no consumo de potência quando utilizou-se 256kB por núcleo, ambas em relação à topologia M0.

Tabela 24 – Aplicação MG - Potência consumida pela rede (mW)

Topologia	***Cache* L2 por núcleo**	**Núcleos**			
		16	32	64	128
C4	64 kB	5,8462	11,6392	24,3639	-
	128 kB	5,7019	11,3061	29,4275	-
	256 kB	5,6554	11,2254	16,9916	-
C8	64 kB	-	-	15,0133	33,5151
	128 kB	-	-	10,7578	33,7381
	256 kB	-	-	10,7390	33,4679
C16	64 kB	-	-	-	21,4478
	128 kB	-	-	-	21,2386
	256 kB	-	-	-	21,2260
M0	64 kB	6,5904	9,4346	16,8917	27,6358
	128 kB	6,4814	11,2992	13,8923	27,5285
	256 kB	6,4533	11,2462	13,4409	27,6354
M1	64 kB	5,2839	9,1432	10,0586	21,0336
	128 kB	5,1933	8,9339	10,1287	21,1373
	256 kB	5,1945	8,9710	10,4363	21,0113
M2	64 kB	5,2782	7,9073	10,3692	21,0280
	128 kB	5,1853	8,9365	10,2965	20,9198
	256 kB	5,2408	8,8626	10,4042	20,9156
M3	64 kB	5,1298	6,6157	10,3530	20,6812
	128 kB	5,0488	8,6268	10,4270	20,7237
	256 kB	5,0383	8,5363	10,4204	20,8400
M4	64 kB	5,1565	8,8147	10,3586	20,7738
	128 kB	5,1147	6,5436	10,3490	20,8226
	256 kB	5,0997	8,7277	10,4335	20,7113
T1	64 kB	5,8914	8,2377	11,6122	22,6137
	128 kB	5,7972	9,6833	11,6099	22,4764
	256 kB	5,7766	9,6332	11,2957	22,6367
T2	64 kB	6,1163	7,6837	11,5936	22,5938
	128 kB	6,0205	9,6280	11,5994	22,6295
	256 kB	6,0488	9,7161	11,7066	22,6141

Continua

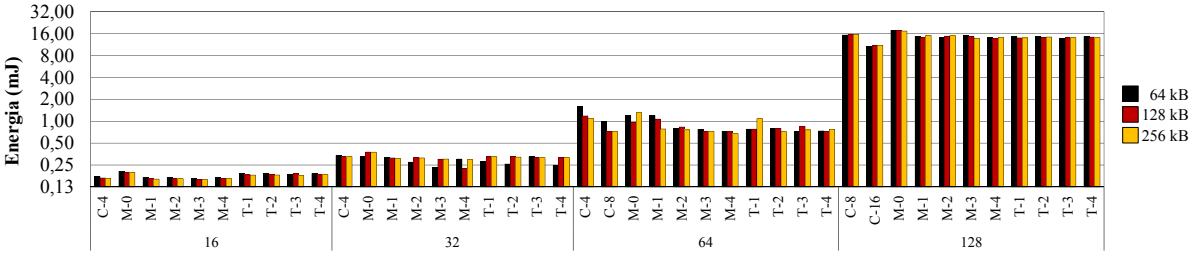
Conclusão

Topologia	*Cache* L2 por núcleo	Núcleos			
		16	32	64	128
T3	64 kB	6,0043	9,8207	11,6926	22,6061
	128 kB	5,5855	9,8544	11,5174	22,6242
	256 kB	5,9128	9,7774	11,6357	22,6417
T4	64 kB	6,1647	7,7383	11,6769	22,4475
	128 kB	6,0336	9,8207	11,6891	22,5929
	256 kB	6,0025	9,7646	11,6115	22,6414
	Melhor resultado	5,0383	6,5436	10,0586	20,6812
	Pior resultado	6,5904	11,6392	29,4275	33,7381

Fonte: Dados da pesquisa

Os resultados de energia para a aplicação MG são apresentados no gráfico da Figura 42 e na Tabela 25. Da mesma forma que na aplicação IS - *Integer Sort*, a topologia do tipo *Cluster* apresenta melhores resultados quando muitos núcleos são usados. Nesse caso, o menor consumo foi de $10,6778mJ$ de energia, com a topologia C16 e 64kB de memória *cache* L2 por núcleo. Com 16 núcleos, a melhor configuração foi aquela com a topologia M3 e 256kB de *cache* L2 por núcleo, com consumo energético de $0,1576mJ$. Já para 32 e 64 núcleos, a topologia M4 apresentou menor consumo energético, respectivamente com 128kB e 256kB de memória *cache* L2 para cada núcleo, sendo $0,2248mJ$ e $0,6767mJ$, também nesta ordem.

Figura 42 – Aplicação MG - Energia consumida pela rede



Núcleos de processamento

Fonte: Dados da pesquisa

Dado que a aplicação também faz comunicações de longa distância, a topologia *Cluster* com menos agrupamentos pode ter saído em desvantagem neste caso, pois os saltos entre núcleos são reduzidos. Outra característica da aplicação se refere aos acessos intensos à memória, mas, ainda assim, reduzir a quantidade de componentes da rede é vantajoso. Como exemplo, a configuração com a topologia M0, que possui uma *cache* L2 por roteador, e apresenta a maioria dos piores resultados. Já na topologia do tipo *Cluster*, a *cache* é conectada por roteador central, e tem tamanho relativo à todos os núcleos daquele agrupamento, que é razoável, não justificando o aumento do tamanho por núcleo, já que os acessos da rede buscam dados suficientes.

Tabela 25 – Aplicação MG - Energia consumida pela rede (mJ)

Topologia	*Cache* L2 por núcleo	Núcleos			
		16	32	64	128
C4	64 kB	0,1726	0,3366	1,5723	-
	128 kB	0,1658	0,3287	1,1812	-
	256 kB	0,1644	0,3250	1,0965	-
C8	64 kB	-	-	0,9942	15,2381
	128 kB	-	-	0,7145	15,3701
	256 kB	-	-	0,7192	15,6500
C16	64 kB	-	-	-	10,6778
	128 kB	-	-	-	10,8540
	256 kB	-	-	-	11,0289
M0	64 kB	0,2047	0,3232	1,1998	17,7018
	128 kB	0,1991	0,3763	0,9573	17,5921
	256 kB	0,1983	0,3751	1,3068	17,3281
M1	64 kB	0,1678	0,3145	1,1944	14,6341
	128 kB	0,1619	0,3122	1,0695	14,2474
	256 kB	0,1603	0,3084	0,7807	15,0336
M2	64 kB	0,1689	0,2738	0,7921	14,1495
	128 kB	0,1646	0,3128	0,8317	14,6051
	256 kB	0,1621	0,3127	0,7580	14,9776
M3	64 kB	0,1635	0,2304	0,7750	14,8470
	128 kB	0,1590	0,3006	0,7182	14,5687
	256 kB	0,1576	0,3017	0,7203	13,7559
M4	64 kB	0,1673	0,3019	0,7228	14,0990
	128 kB	0,1631	0,2248	0,7171	13,5785
	256 kB	0,1622	0,2961	0,6767	14,2644
T1	64 kB	0,1887	0,2799	0,7774	14,5257
	128 kB	0,1828	0,3249	0,7776	13,9128
	256 kB	0,1820	0,3241	1,0809	13,9504
T2	64 kB	0,1900	0,2580	0,7877	14,4479
	128 kB	0,1841	0,3271	0,7889	13,9651
	256 kB	0,1821	0,3213	0,7156	14,3405
T3	64 kB	0,1854	0,3253	0,7223	13,8023
	128 kB	0,1902	0,3150	0,8440	14,0128
	256 kB	0,1795	0,3150	0,7582	14,0100
T4	64 kB	0,1898	0,2493	0,7344	14,5566
	128 kB	0,1844	0,3163	0,7260	14,3036
	256 kB	0,1834	0,3151	0,7757	14,0525
	Melhor resultado	0,1576	0,2248	0,6767	10,6778
	Pior resultado	0,2047	0,3763	1,5723	17,7018

Fonte: Dados da pesquisa

Já que, na aplicação MG, quando processada com mais de 64 núcleos, prevalecem as comunicações do tipo *broadcast*, a topologia *Cluster* é favorecida. Isso ocorre porque esta topologia diminui o número de saltos entre os núcleos que se comunicam, e com um padrão de comunicação *broadcast*, a atividade de roteamento e uso de links acaba sendo reduzido, diminuindo o consumo energético.

5.6 Considerações finais sobre os resultados

Com a análise das cinco aplicações, é possível concluir que o simples aumento da quantidade de núcleos não é uma boa estratégia, sem ser acompanhada de uma análise específica dos objetivos que se deseja alcançar. Faz-se necessário o uso de estratégias para melhor utilizar os vários núcleos disponíveis, como mapeamento de processos. De toda forma, ainda que uso das redes-em-chip favoreça o aumento da quantidade de núcleos para ganho em poder de processamento, considerando-se as estratégias necessárias para isso, é necessário ter atenção com o consumo energético.

Os resultados permitiram identificar, dentre as configurações de topologia, quantidade de núcleos e tamanho de *caches*, quais arquiteturas oferecem melhores opções em termos de consumo energético. Toma-se como princípio o fato de que quanto maior o consumo, e também quanto maior o tempo de processamento da aplicação, pior será o resultado. Dessa forma, nessa dissertação, calculou-se o produto dessas duas métricas, o consumo de energia, dado por $Energia = Tempo \times Potencia$.

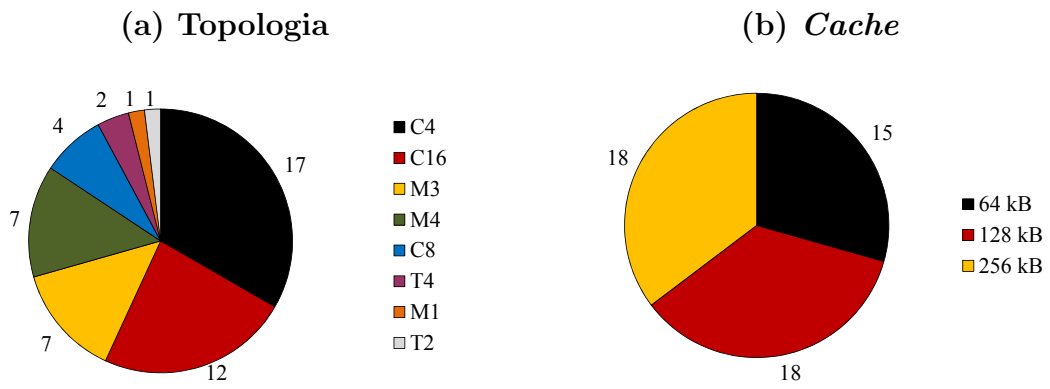
Observando os resultados, percebe-se que a estratégia de agrupamentos, através do uso das topologias do tipo *Cluster*, é a que apresenta melhores condições de compor um processador *many-core* com baixo consumo energético. A Tabela 26 apresenta os três melhores resultados obtidos com cada quantidade de núcleos, por aplicação simulada. Dentre esses resultados, é possível notar que as topologias do tipo *Cluster* se destacam. Pelo gráfico da Figura 43a, tem-se que somente as topologias C4 e C16 compreendem mais da metade dos melhores resultados.

As topologias M3 e M4, do tipo *Mesh*, aparecem como alternativa intermediária, destacando-se principalmente quando simuladas as aplicações IS e MG. Conforme destacado nas Seções 5.5 e 5.4, o posicionamento das *caches* em roteadores com mais de um salto de distância entre eles pode ter favorecido essas aplicações. O uso da rede torna-se mais distribuído, e para as aplicações que fazem acessos intensos e aleatórios à memória, ter caminhos disponíveis para alcançar a *cache* é importante. Com o aumento da quantidade de núcleos, até 128, a topologia *Cluster* aparece novamente como melhor opção, em todos os casos.

Com relação aos tamanhos de *caches*, a de 64 kB é a que aparece menos vezes

dentre os melhores resultados. Esse número, por outro lado, é ligeiramente próximo das que aparecem mais vezes. Pelo gráfico da Figura 43b, é possível notar uma certa homogeneidade dentre os três melhores resultados de cada grupo. Avaliando-se, pela Tabela 26, somente os primeiros colocados, constata-se que mais da metade dos casos são com as configurações com *caches* de 256 kB por núcleo (nove ocorrências).

Figura 43 – Distribuição dos três melhores resultados



Fonte: Dados da pesquisa

É importante ressaltar que, em alguns casos, a alteração de topologia influencia mais nos melhores resultados do que a variação no tamanho da *cache* por núcleo. É o caso, por exemplo, quando usados 128 núcleos, em que a topologia C16 aparece em todos os melhores resultados, com os três tamanhos de *cache* configurados. Neste caso em específico, é digno de nota que usar 64 kB de *cache* por núcleo, com as aplicações IS e MG, apresenta melhores resultados. Esse comportamento ressalta a necessidade de um estudo detalhado do comportamento das *caches*, compreendendo análises de protocolos de coerência, da alocação de dados em *cache*, largura de banda no acesso à memória, dentre outros, o que é sugestão para trabalhos futuros. Além disso, associar essa análise com a criação de aplicações adequadas para processadores *many-core*.

De maneira geral, pode-se considerar correta a proposta do processador MPPA-256 de utilizar a abordagem de agrupamentos. É importante, ainda, ressaltar que o uso de mais núcleos certamente irá aumentar o consumo energético, portanto, os melhores resultados foram apresentados por quantidade de núcleos para abstrair esse comportamento. A arquitetura por agrupamentos seria a melhor, conforme o escopo desta dissertação e da exploração de espaço de projeto definida, para atingir o objetivo de redução em consumo energético e aumento de performance.

Tabela 26 – Resumo dos resultados - Consumo energético (mJ)

		Núcleos											
		16			32			64			128		
		Topologia	L2	Energia	Topologia	L2	Energia	Topologia	L2	Energia	Topologia	L2	Energia
CG	1º	C4	256 kB	0,2704	C4	256 kB	0,4772	-	-	-	C16	256 kB	36,7323
	2º	C4	128 kB	0,2719	C4	128 kB	0,4794	-	-	-	C16	128 kB	36,8026
	3º	C4	64 kB	0,2817	C4	64 kB	0,4799	-	-	-	C16	64 kB	38,6543
EP	1º	C4	128 kB	1,9531	C4	128 kB	2,7704	C8	128 kB	5,0123	-	-	-
	2º	C4	256 kB	1,9673	C4	64 kB	2,7839	M3	256 kB	5,6721	-	-	-
	3º	C4	64 kB	1,9743	C4	256 kB	2,7846	M4	128 kB	5,7683	-	-	-
FT	1º	C4	256 kB	0,5661	C4	256 kB	0,8491	M4	64 kB	1,5749	C16	256 kB	6,0936
	2º	C4	128 kB	0,5744	M4	128 kB	0,9626	C8	256 kB	1,5794	C16	128 kB	6,1031
	3º	C4	64 kB	0,6209	C4	128 kB	0,9768	C8	64 kB	1,5896	C16	64 kB	6,3319
IS	1º	M3	256 kB	0,1520	M4	128 kB	0,2242	-	-	-	C16	64 kB	16,2197
	2º	M3	128 kB	0,1533	M3	64 kB	0,2303	-	-	-	C16	128 kB	16,3999
	3º	M4	256 kB	0,1558	T4	64 kB	0,2518	-	-	-	C16	256 kB	16,4123
MG	1º	M3	256 kB	0,1576	M4	128 kB	0,2248	M4	256 kB	0,6767	C16	64 kB	10,6778
	2º	M3	128 kB	0,1590	M3	64 kB	0,2304	C8	128 kB	0,7145	C16	128 kB	10,8540
	3º	M1	256 kB	0,1603	T4	64 kB	0,2493	T2	256 kB	0,7156	C16	256 kB	11,0289

Fonte: Dados da pesquisa

6 CONCLUSÕES E TRABALHOS FUTUROS

O aumento da demanda por alto poder de processamento e redução no consumo energético são dois desafios que persistem no estudo e desenvolvimento de arquiteturas paralelas. Ter técnicas de baixo custo e com boa flexibilidade para conduzir estudos para ultrapassar esses desafios torna-se de grande importância, seja para o bom andamento de pesquisas, ou para competitividade de mercado. Este trabalho apresentou duas abordagens que podem ser usadas em conjunto para atingir esse objetivo: A simulação de sistemas completos, e a DSE, ou exploração de espaço de projeto. A hipótese era que a simulação pudesse fornecer meios para a avaliação de processadores, elementos de memória e redes de interconexão, permitindo análises do comportamento de aplicações simuladas em diversas arquiteturas, explorando então as várias alternativas e definindo aquelas que proporcionam a melhor eficiência energética.

Alguns trabalhos que usam as técnicas de simulação e DSE podem ser encontrados na literatura, mas esses não esgotam o panorama atual de pesquisa em processadores com redes-em-chip, abordando a configuração de topologias diversas, tamanhos diversos de *cache* L2 e quantidade de núcleos, em um único experimento. A DSE mostrou ser uma técnica que auxilia o pesquisador nas etapas iniciais de pesquisa, organizando as ideias de projetos, evitando grandes desvios que poderiam prejudicar a condução dos trabalhos. Já na simulação de sistemas completos, percebe-se um potencial de opções extremamente extenso, o que requer paciência e determinação. O simulador *Gem5* foi estudado e utilizado, permitindo abranger o escopo da DSE de maneira satisfatória.

Como arquitetura alvo de estudo dessa dissertação, o processador *many-core* MPPA-256 foi selecionado. A arquitetura desse processador abrange diversos aspectos que podem ser avaliados, dentre eles: grande quantidade de núcleos, redes-em-chip e memórias compartilhadas e distribuídas. Também foram verificadas quais as limitações desse processador, em trabalhos correlatos. A partir dessa arquitetura, foi possível definir e explorar um espaço de projeto com objetivos específicos, que resultou na configuração de 531 opções diferentes de arquitetura, variando em quantidade de núcleos, tamanho de *cache* L2 por núcleo e topologias, propondo diferentes organizações dos roteadores e componentes do processador.

Com os resultados, foi possível identificar configurações de arquiteturas que apresentam vantagens para determinadas aplicações, em termos de potência consumida, tempo simulado, latência de rede e taxa de faltas. Considerou-se como melhor, pelos resultados apurados, a topologia do tipo *Cluster* com roteadores centrais para *caches*, agrupando, por meio de outros roteadores, os núcleos. Esse resultado, que apontou ganhos em consumo energético em relação à topologias tradicionais como a *Mesh*, reforça

a importância das técnicas apresentadas neste trabalho como alternativa para vencer os desafios estabelecidos.

Um ponto negativo constatado durante os trabalhos, é o tempo de execução do *Gem5* para simulação. Cada simulação pode demorar dias, e 531 simulações demandaram bastante tempo. O tempo médio gasto para simular todas as aplicações, em uma configuração com 16 núcleos, foi de 23 horas e 4 minutos. Para 32 núcleos, o tempo médio gasto foi de 31 horas e 51 minutos. Com 64 núcleos, o tempo médio foi de 32 horas e 42 minutos. Por fim, na configuração com 128 núcleos, o tempo médio gasto foi de 165 horas e 53 minutos. Pouca variação no tempo gasto para a simulação foi observada quando alterou-se a topologia ou tamanho de *cache* L2. Para otimizar o tempo das simulações, optou-se por executar várias arquiteturas paralelamente. Com a execução do projeto, detalhado no Capítulo 3, e avaliação dos resultados apresentados, foi possível alcançar os objetivos inicialmente propostos.

Como trabalhos futuros, sugere-se, em um primeiro momento, o estudo mais aprofundado das memórias *cache* L2 em processadores *many-core*. Propõe-se um estudo do comportamento dos protocolos de coerência, dos efeitos das alterações de tamanho de linhas de *cache*, da influência das alterações de latência em *caches* de tamanhos diferentes, bem como o acompanhamento mais detalhado da forma como os dados são compartilhados nessas memórias. Outra sugestão de trabalho futuro, é a criação de um pacote de aplicações apropriados para o estudo de processadores *many-core*. Neste contexto, o estudo de diferentes classes do NPB, para estudos de escalabilidade de aplicações, com a variação do tamanho da carga de trabalho, também é uma proposta de trabalho futuro. O conjunto de aplicações usado neste trabalho foi suficiente para alcançar os objetivos, entretanto, os resultados para uma quantidade grande de núcleos (128, por exemplo) mostraram-se insatisfatórios. O estudo de técnicas de mapeamento de processos, em conjunto com o estabelecimento desse novo pacote de aplicações, também pode ser agregado nesse trabalho, ampliando a aplicabilidade de programas para esse fim.

REFERÊNCIAS

AGARWAL, N. et al. GARNET: A detailed on-chip network model inside a full-system simulator. In: IEEE INTERNATIONAL SYMPOSIUM ON PERFORMANCE ANALYSIS OF SYSTEMS AND SOFTWARE (ISPASS), 2009, Boston. **Proceedings...** Boston: IEEE, 2009. p. 33–42.

ASANOVIC, K. et al. A view of the parallel computing landscape. **Communications of the ACM**, New York, v. 52, n. 10, p. 56–67, 2009. Disponível em: <http://doi.acm.org/10.1145/1562764.1562783>. Acesso em: 04 dez. 2014.

AVELAR, C. P. **Avaliação de abordagens de mapeamento de processos em redes-em-chip para aplicações paralelas**. 2014. 42 f. Dissertação (Mestrado) — Pontifícia Universidade Católica de Minas Gerais, Programa de Pós-Graduação em Informática, Belo Horizonte, 2014.

BAILEY, D. H. et al. The NAS parallel benchmarks. In: ACM/IEEE CONFERENCE ON SUPERCOMPUTING, 1991, Albuquerque. **Proceedings**. New York: ACM, 1991. p. 158–165.

BENINI, L.; MICHELI, G. D. Networks on chips: a new soc paradigm. **Computer**, Los Alamitos, v. 35, n. 1, p. 70–78, Jan 2002. Disponível em: <http://dx.doi.org/10.1109/2.976921>. Acesso em: 12 set. 2014.

BINKERT, N. et al. The gem5 simulator. **ACM SIGARCH Computer Architecture News**, New York, v. 39, n. 2, p. 1–7, 2011. Disponível em: <http://doi.acm.org/10.1145/2024716.2024718>. Acesso em: 26 set. 2013.

BINKERT, N. L. et al. The M5 simulator: Modeling networked systems. **IEEE Micro**, Los Alamitos, v. 26, n. 4, p. 52–60, 2006. Disponível em: <http://dx.doi.org/10.1109/MM.2006.82>. Acesso em: 26 set. 2013.

BJERREGAARD, T.; MAHADEVAN, S. A survey of research and practices of network-on-chip. **ACM Computing Surveys (CSUR)**, New York, v. 38, n. 1, 2006. Disponível em: <http://doi.acm.org/10.1145/1562764.1562783>. Acesso em: 04 dez. 2014.

CAMACHO, J. et al. A power-efficient network on-chip topology. In: INTERNATIONAL WORKSHOP ON INTERCONNECTION NETWORK ARCHITECTURE: ON-CHIP, MULTI-CHIP, 5TH., 2011, Heraklion. **Proceedings...** New York: ACM, 2011. (INA-OCMC '11), p. 23–26.

CASTRO, M. et al. Analysis of computing and energy performance of multicore, NUMA, and manycore platforms for an irregular application. In: WORKSHOP ON IRREGULAR APPLICATIONS: ARCHITECTURES AND ALGORITHMS, 3RD., 2013, Denver. **Proceedings...** Denver: ACM, 2013. p. 5:1–5:8.

CHEN, L.; WANG, R.; PINKSTON, T. M. Efficient implementation of globally-aware network flow control. **Journal of Parallel and Distributed Computing**, Orlando, v. 72, n. 11, p. 1412–1422, 2012. Disponível em: <http://dx.doi.org/10.1016/j.jpdc.2012.02.004>. Acesso em: 21 abr. 2014.

CHRYSOS, G.; ENGINEER, S. P. Intel xeon phi coprocessor (codename knights corner). In: HOT CHIPS SYMPOSIUM, 24TH., 2012, Cupertino. **Proceedings...** Cupertino, 2012.

DANESHTALAB, M. et al. A generic adaptive path-based routing method for MPSoCs. **Journal of Systems Architecture**, New York, v. 57, n. 1, p. 109–120, 2011. Disponível em: <http://dx.doi.org/10.1016/j.sysarc.2010.08.002>. Acesso em: 18 abr. 2014.

DINECHIN, B. de et al. A clustered manycore processor architecture for embedded and accelerated applications. In: IEEE HIGH PERFORMANCE EXTREME COMPUTING CONFERENCE (HPEC), 2013, Waltham. **Proceedings...** Waltham: IEEE, 2013. p. 1–6.

FLEIG, T.; MATTES, O.; KARL, W. Evaluation of adaptive memory management techniques on the tilera TILE-Gx platform. In: INTERNATIONAL CONFERENCE ON ARCHITECTURE OF COMPUTING SYSTEMS (ARCS), 27TH., 2014, Lueback. **Proceedings...** Luebeck: VDE, 2014. p. 1–8.

FLYNN, M. Very high-speed computing systems. **Proceedings of the IEEE**, San Francisco, v. 54, n. 12, p. 1901–1909, Dec 1966. Disponível em: <http://dx.doi.org/10.1109/PROC.1966.5273>. Acesso em: 12 dez. 2014.

FRANCESQUINI, E. et al. On the energy efficiency and performance of irregular application executions on multicore, NUMA and manycore platforms. **Journal of Parallel and Distributed Computing**, Orlando, v. 76, p. 32–48, 2015. Disponível em: <http://dx.doi.org/10.1016/j.jpdc.2014.11.002>. Acesso em: 30 mar. 2015.

FRANCESQUINI, E.; GOLDMAN, A.; MÉHAUT, J.-F. Improving the performance of actor model runtime environments on multicore and manycore platforms. In: WORKSHOP ON PROGRAMMING BASED ON ACTORS, AGENTS AND DECENTRALIZED CONTROL, 2013, Indianapolis. **Proceedings...** New York: ACM, 2013. p. 109–114.

FREITAS, H. C. **Arquitetura de NoC programável baseada em múltiplos clusters de cores para suporte a padrões de comunicação coletiva**. 2009. 63 f. Tese (Doutorado) — Universidade Federal do Rio Grande do Sul, Programa de Pós-Graduação em Computação, Porto Alegre, 2009.

FREITAS, H. C. d.; ALVES, M. A. Z.; NAVAUX, P. O. A. Noc e nuca: Conceitos e tendências para arquiteturas de processadores many core. In: ESCOLA REGIONAL DE ALTO DESEMPENHO (ERAD), IX., 2009, Caxias do Sul. **Anais...** Caxias do Sul: SBC, 2009. p. 364–366.

GUERRE, A. et al. Hierarchical network-on-chip for embedded many-core architectures. In: ACM/IEEE INTERNATIONAL SYMPOSIUM IN NETWORKS-ON-CHIP (NOCS), 4TH., 2010, Grenoble. **Proceedings...** Grenoble: IEEE, 2010. p. 189–196.

HENNESSY, J. L.; PATTERSON, D. A. **Computer Architecture**: a quantitative approach. 4. ed. San Francisco: M. Kaufmann, 2007.

IVANOV, L.; NUNNA, R. Modeling and verification of cache coherence protocols. In: IEEE INTERNATIONAL SYMPOSIUM ON CIRCUITS AND SYSTEMS (ISCAS), 2001, Sydney. **Proceedings...** Sydney: IEEE, 2001. v. 5, p. 129–132.

JAIN, R. **The art of computer systems performance analysis**: techniques for experimental design, measurement, simulation, and modeling. New York: Wiley, 1991.

KAHNG, A. B. et al. ORION 2.0: A fast and accurate noc power and area model for early-stage design space exploration. In: DESIGN, AUTOMATION AND TEST IN EUROPE CONFERENCE EXHIBITION (DATE), 2009, Nice. **Proceedings...** Nice: IEEE, 2009. p. 423–428.

KANG, E.; JACKSON, E.; SCHULTE, W. An approach for effective design space exploration. In: MONTEREY CONFERENCE ON FOUNDATIONS OF COMPUTER SOFTWARE: MODELING, DEVELOPMENT, AND VERIFICATION OF ADAPTIVE SYSTEMS, 2011, Redmond. **Proceedings...** Berlin: Springer-Verlag, 2011. p. 33–54.

KIM, H. et al. Reducing network-on-chip energy consumption through spatial locality speculation. In: IEEE/ACM INTERNATIONAL SYMPOSIUM ON NETWORKS-ON-CHIP (NOCS), 5TH., 2011, Pittsburgh. **Proceedings...** Pittsburgh: IEEE, 2011. p. 233–240.

KUNZLI, S. **Efficient design space exploration for embedded systems**. 2006. 98 f. Tese (Doutorado) — Swiss Federal Institute of Technology Zurich, Zurich, 2006.

LI, X.; HAMMAMI, O. Nocdex: Network on chip design space exploration through direct execution and options selection through principal component analysis. In: INTERNATIONAL SYMPOSIUM ON INDUSTRIAL EMBEDDED SYSTEMS, 6., 2006, Antibes Juan-Les-Pins. **Proceedings...** Antibes Juan-Les-Pins: IEEE, 2006. p. 1–4.

MAGNUSSON, P. S. et al. Simics: A full system simulation platform. **Computer**, Los Alamitos, v. 35, n. 2, p. 50–58, fev. 2002. Disponível em: <http://dx.doi.org/10.1109/2.982916>. Acesso em: 12 set. 2013.

MARTIN, M. M. et al. Multifacet's general execution-driven multiprocessor simulator (GEMS) toolset. **ACM SIGARCH Computer Architecture News**, New York, v. 33, n. 4, p. 92–99, 2005. Disponível em: <http://dx.doi.org/10.1145/1105734.1105747>. Acesso em: 26 set. 2013.

MATSUTANI, H. et al. Ultra fine-grained run-time power gating of on-chip routers for CMPs. In: ACM/IEEE INTERNATIONAL SYMPOSIUM IN NETWORKS-ON-CHIP (NOCS), 4TH., 2010, Grenoble. **Proceedings...** Grenoble, 2010. p. 61–68.

MURALIDHARA, S. P.; KANDEMIR, M. Communication based proactive link power management. **High-Performance Embedded Architectures and Compilers**, Berlin, v. 5409, p. 135–154, 2011. Disponível em: <http://dx.doi.org/10.1007/978-3-540-92990-1_16>. Acesso em: 16 abr. 2014.

OLIVEIRA, P. A. C. d. **Avaliação de desempenho e caracterização de cargas de trabalho paralelas para redes-em-chip sem fio**. 2012. 92 f. Dissertação (Mestrado) — Pontifícia Universidade Católica de Minas Gerais, Programa de Pós-Graduação em Informática, Belo Horizonte, 2012.

PATTERSON, D.; HENNESSY, J. **Computer Organization and Design**: the hardware/software interface. 4. ed. rev. Waltham: M. Kaufmann, 2012.

RENAU, J. et al. **SESC simulator**. January 2005. Disponível em: <http://sesc.sourceforge.net>. Acesso em: 26 set. 2013.

ROSE, C. D.; NAVAUX, P. Fundamentos de processamento de alto desempenho. In: ESCOLA REGIONAL DE ALTO DESEMPENHO (ERAD), II., 2002, São Leopoldo. **Anais...** São Leopoldo: SBC, 2002. p. 3–29.

SATO, L. M.; MIDORIKAWA, E. T.; SENGER, H. Introdução a programação paralela e distribuída. In: JORNADA DE ATUALIZAÇÃO EM INFORMÁTICA, XV., 1996, Recife. **Anais...** Recife: SBC, 1996. p. 1–56.

SHALF, J.; DOSANJH, S.; MORRISON, J. Exascale computing technology challenges. In: INTERNATIONAL CONFERENCE ON HIGH PERFORMANCE COMPUTING FOR COMPUTATIONAL SCIENCE, 9TH., 2011, Berkley. **Proceedings...** Berlin: Springer-Verlag, 2011. p. 1–25.

SIMON, H. D. Barriers to exascale computing. **High Performance Computing for Computational Science - VECPAR 2012**, Kope, v. 7851, 2012. Disponível em: <http://dx.doi.org/10.1007/978-3-642-38718-0_1>. Acesso em: 09 set. 2014.

SOUZA, M. A. et al. Avaliação do consumo energético em arquiteturas multi-core com memória cache compartilhada. In: WORKSHOP EM DESEMPENHO DE SISTEMAS COMPUTACIONAIS E DE COMUNICAÇÃO - WPERFORMANCE, XIII., 2014, Brasília. **Anais...** Brasília, 2014. p. 1812–1824.

SUH, T.; BLOUGH, D. M.; LEE, H.-H. S. Supporting cache coherence in heterogeneous multiprocessor systems. In: DESIGN, AUTOMATION AND TEST IN EUROPE CONFERENCE EXHIBITION (DATE), 2004, Paris. **Proceedings...** Paris: IEEE, 2004. v. 2, p. 1150–1155.

TRIVIÑO, F. et al. Virtualizing network-on-chip resources in chip-multiprocessors. **Microprocessors and Microsystems**, Amsterdam, v. 35, n. 2, p. 230–245, 2011. Disponível em: <http://dx.doi.org/10.1016/j.micpro.2010.10.001>. Acesso em: 15 abr. 2014.

UDIPI, A. N.; MURALIMANOHAR, N.; BALASUBRAMONIAN, R. Towards scalable, energy-efficient, bus-based on-chip networks. In: IEEE INTERNATIONAL SYMPOSIUM ON HIGH-PERFORMANCE COMPUTER ARCHITECTURE (HPCA), 16TH., 2010, Bangalore. **Proceedings...** Bangalore: IEEE, 2010. p. 1–12.

XU, T. C. et al. Optimal placement of vertical connections in 3d network-on-chip. **Journal of Systems Architecture**, New York, v. 59, n. 7, p. 441–454, 2013. Disponível em: <http://dx.doi.org/10.1016/j.sysarc.2013.05.002>. Acesso em: 16 abr. 2014.

YI, J.; LILJA, D. Simulation of computer architectures: simulators, benchmarks, methodologies, and recommendations. **IEEE Transactions on Computers**, Los Alamitos, v. 55, n. 3, p. 268–280, March 2006. Disponível em: <http://dx.doi.org/10.1109/TC.2006.44>. Acesso em: 09 fev. 2015.

ZEFERINO, C. A. **Redes-em-chip**: Arquiteturas e modelos para avaliação de área e desempenho. 2003. 121 f. Tese (Doutorado) — Universidade Federal do Rio Grande do Sul, Programa de Pós-Graduação em Computação, Porto Alegre, 2003.