Avaliação de Topologias de Redes-em-Chip usando Simulação de Sistemas Completos e Aplicações Paralelas

Daniel A. S. Carmo, Matheus A. Souza, Henrique C. Freitas

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

{daniel.carmo,matheus.alcantara}@sga.pucminas.br, cota@pucminas.br

Abstract. *The Networks-on-Chip approach is an alternative to achieve high-performance computing. In spite of using traditional interconnection systems, this strategy uses routers to enable the communication among the diverse cores in a many-core processor. However, with this approach, the processor performance may be compromised if the interconnection design was not properly planned. The bandwitdh degradation as well as unbalanced loads between the components of the processor could degrade the performance of those architectures. Our work aims to present how the processor performance may be affected by the network-on-chip topology and the design to interconnect memories. The results showed that the cluster topology architectures have better performance than the other architectures.*

Resumo. *A abordagem de Redes-em-Chip é uma alternativa na busca por poder de processamento. Ao invés de usar sistemas de interconexão tradicionais, essa estratégia utiliza roteadores para permitir a comunicação entre os núcleos de um processador* many-core. *Porém, o desempenho do processador pode ficar comprometido caso o projeto de interconexão não esteja bem planejado. A degradação da largura de banda e a presença de desbalanceamento de cargas entre os componentes do processador podem reduzir o desempenho destas arquiteturas. Este trabalho tem como objetivo mostrar como o desempenho do processador pode ser afetado pela topologia da rede-em-chip e o projeto para interconectar as memórias. Os resultados mostraram que as arquiteturas organizadas em cluster apresentaram desempenho melhor em relação as outras arquiteturas.*

1. Introdução

A demanda por alto desempenho das aplicações paralelas aumenta a cada dia, e o volume de dados processados por estas aplicações também aumenta. Dessa forma, a área de Ciência da Computação tem o desafio de propor processadores que sejam capazes de atender a tal demanda. Nas últimas décadas, diversas arquiteturas de processadores foram propostas para atender à demanda por desempenho, com alto poder de processamento. É o caso das arquiteturas paralelas, que contemplam em um único *chip* vários núcleos de processamento, como as *Graphics Processing Units (GPUs)* e os processadores *multi* e

many-core [Asanovic et al. 2009, Shalf et al. 2010, Simon 2012] como estratégia também para o aumento de escalabilidade dos *Clusters* e *Grids* computacionais.

Uma arquitetura de processador *multi-core* contempla vários núcleos em um único *chip* (e.g. *quad-cores* e *octa-cores*), que normalmente utilizam de estratégias de interconexão mais simples, como barramentos e chaves *crossbar*. Já as arquiteturas *many-core*, contemplam um alto número de núcleos (e.g. 256 núcleos), e as estratégias de interconexão tradicionais tornam-se inadequadas para este tipo de processador, graças à restrições inerentes à essas tecnologias, como a concorrência por recursos, largura de banda, atenuação de sinal e alta latência das longas interconexões [Ho et al. 2001].

Para superar esse problema de interconexão nas arquiteturas *many-core*, propõe-se o uso das redes-em-chip ou *networks-on-chip* (NoCs) [Benini and Micheli 2002]. Uma NoC é uma rede de interconexão baseada no uso de roteadores para interconectar o alto número de núcleos e componentes (unidades de memória, por exemplo), permitindo a redução do comprimento dos circuitos eletrônicos, mitigando os problemas das interconexões tradicionais e aumentando a escalabilidade do *chip* [Freitas et al. 2009].

As NoCs podem ter configurações diferentes, isto é, apresentar formas de organização (topologias), quantidades de componentes e protocolos diferentes, dentre outras características. Como exemplo, memórias *cache* podem ser conectadas à NoC de maneira estratégica, para favorecer o acesso à memória dos núcleos de processamento durante a execução de alguma aplicação.

Com a descentralização do processamento, a comunicação entre os núcleos se torna uma peça chave em qualquer NoC. Tanto a transmissão de dados entre processadores e processadores e controladores de memória precisam ser bem formulados para que nenhum núcleo fique ocioso ou sobrecarregado.

Ainda que seja uma proposta excelente em termos de escalabilidade, as NoCs estão em constante avaliação e evolução, no sentido de resolver problemas ainda existentes, como a perda de largura de banda, desbalanceamento de cargas e consumo de potência. Dessa maneira, explorar as diversas possibilidades de composição e o comportamento das NoCs é essencial para permitir a evolução dessas arquiteturas.

Observados esses aspectos, este artigo propõe a análise de diversas topologias de NoCs, avaliando o comportamento da rede em relação à latência e largura de banda das propostas, bem como o consumo de potência dos componentes da NoC. Para permitir esta análise, usa-se o simulador de sistemas completos *Gem5* [Binkert et al. 2011] em conjunto com o pacote de aplicações *CAP Bench* [Souza et al. 2016], como um ambiente preparado para a análise de arquiteturas *many-core* baseadas em NoCs.

A organização do artigo está descrita à seguir. Na Seção 2, é apresentada como uma relação de trabalhos correlatos. Em seguida, na Seção 3, são apresentadas as topologias propostas no experimento. A Seção 4 descreve os procedimentos metodológicos do trabalho, enquanto na Seção 5 discute-se os resultados obtidos. Por fim, na Seção 6 são realizadas considerações finais e propostos trabalhos futuros.

2. Trabalhos Relacionados

A organização de uma arquitetura de uma NoC é de grande importância, uma vez que um sistema que possui características semelhantes podem apresentar resultados bem di-

ferentes dependendo da forma como seus componentes estão dispostos. A presença de desbalanceamento de cargas de trabalho e a degradação da largura de banda são problemas que podem impactar no desempenho da NoC. Além disso, o consumo de potência também pode ser um ponto negativo de uma NoC inadequada para determinada aplicação. Dessa forma, a avaliação das arquiteturas apresentadas neste trabalho pode evidenciar quais organizações de NoCs apresentam os melhores resultados, como tendência para novas arquiteturas. De maneira similar, os trabalhos a seguir apresentam propósitos semelhantes.

Em [Freitas et al. 2008] é proposta uma arquitetura de NoC que suporta múltiplos *Clusters* de núcleos de processamento, por meio de roteadores programáveis e de topologias reconfiguráveis, demonstrando esta abordagem como alternativa para a construção de novas topologias de NoCs.

De forma similar, uma topologia de rede hierárquica, também baseada em *Clusters*, é apresentada em [Udipi et al. 2010]. A proposta é utilizar em conjunto a NoC, com roteadores, e barramentos tradicionais, com cada tipo de interconexão em um nível da rede. Ainda que os autores tenham usado simulação, não utilizaram cargas de trabalhos próprias para *many-cores*.

Já em [Camacho et al. 2011], foi proposta uma NoC denominada *NR-Mesh*. O princípio dessa NoC, baseada em uma topologia *Mesh*, é o uso de roteadores diferentes do que o núcleo está conectado para enviar e receber mensagens pela rede.

Por fim, uma topologia de NoC 3D é apresentada em [Xu et al. 2013]. Nesse trabalho, novos algoritmos de roteamento e disposições dos roteadores da NoC foram propostos, a fim de projetar a topologia 3D.

Apesar dos trabalhos mencionados serem semelhantes ao trabalho proposto, o uso do *CAP Bench* diferencia este dos demais, que, em conjunto com o *Gem5* permite a analise da largura de banda, latência, requisições de leitura/escrita, quantidade de *bytes* lidos/escritos e o consumo de potência das simulações utilizando diferentes abordagens. Essas métricas não são completamente abordadas nos trabalhos, e a simulação de sistemas completos também não, o que reforça a contribuição deste artigo.

3. Propostas de arquiteturas

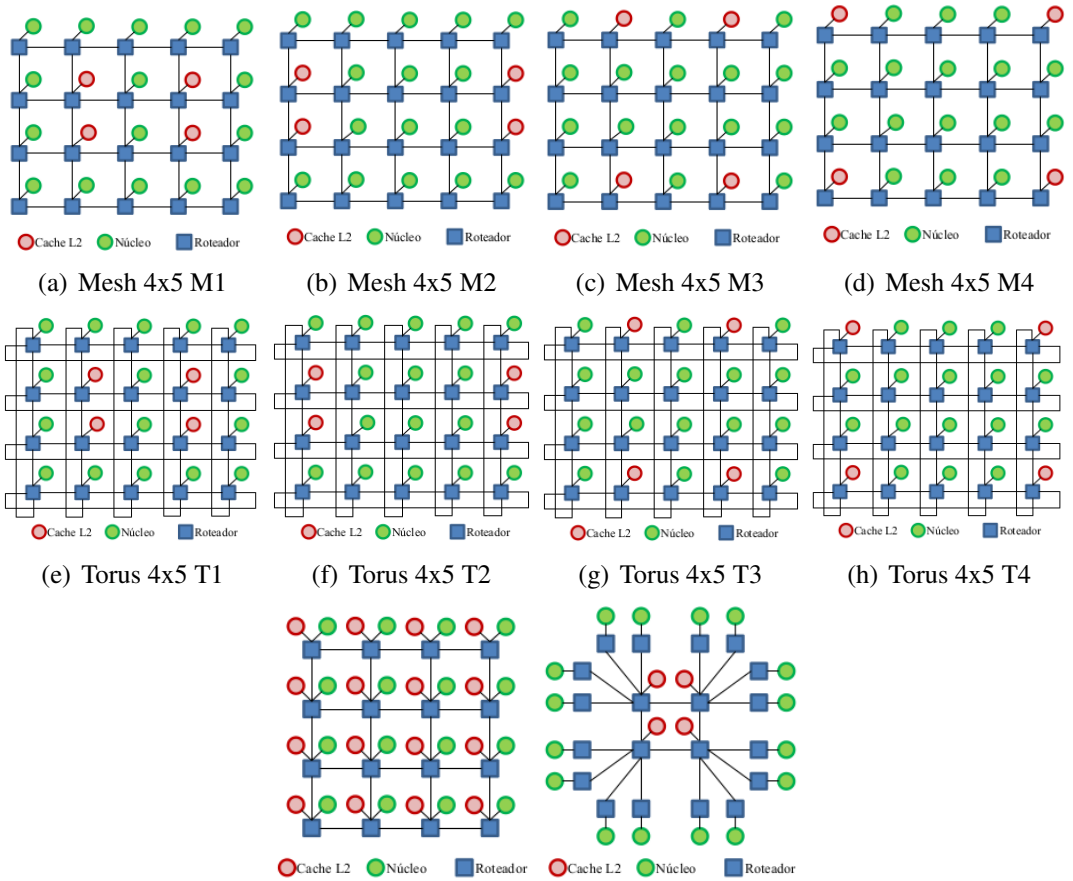
Este trabalho baseia-se nas arquiteturas propostas em [Souza 2015], que variam em números de núcleos (16 ou 32 núcleos) e organizações de topologia (Malha, *Torus* e *Cluster*). O número de roteadores dentro de cada arquitetura varia dentro da rede. A forma as quais foram escolhidas a posição das memórias foi alterada de forma a evidenciar qual a influência da sua posição no desempenho do sistema como um todo.

Com exceção da arquitetura denominada M0, que possui uma organização em Malha com 16 ou 32 caches L2 para organizações com 16 ou 32 núcleos, respectivamente, todas as outras arquiteturas de 16 ou 32 núcleos possuem 4 ou 8 *caches* L2, respectivamente. O número de roteadores também é diferente nesta arquitetura. A M0 possui um número de roteadores iguais ao número de núcleos. Todas as demais possuem um número de roteadores igual a soma do número de núcleos com o número de *caches* L2..

A posição das *caches* L2 é o que difere as arquiteturas que possuem o mesmo número de núcleos e a mesma topologia. Com exceção da arquitetura M0, cada cache

possui 64 kB de memória. Na arquitetura M0, cada *cache* possui 16 kB de memória. Assim, cada simulação possui 256 kB de memória ao todo.

A arquitetura M0 possui um controlador de memória e um processador para cada roteador, sendo adotada como arquitetura base para comparações nos resultados pois é uma arquitetura tradicional. As arquiteturas CL são organizadas em *Cluster*, onde há um roteador para cada núcleo e cada 4 roteadores ligados a núcleos estão conectados a um roteador ligados a uma memória *cache* L2. A Figura 1 mostra essas duas arquiteturas.



(a) Mesh 4x5 M1 (b) Mesh 4x5 M2 (c) Mesh 4x5 M3 (d) Mesh 4x5 M4

(e) Torus 4x5 T1 (f) Torus 4x5 T2 (g) Torus 4x5 T3 (h) Torus 4x5 T4

(i) Mesh 4x4 M0 (j) Mesh 2x2 Cluster

Figura 1. Arquiteturas Utilizadas
Fonte:[Souza 2015]

4. Metodologia de Avaliação

Para avaliar as arquiteturas propostas, foi escolhido o simulador *Gem5* [Binkert et al. 2011]. É uma plataforma modular para pesquisas em sistemas de computadores. Com esta ferramenta, é possível planejar, criar, simular, analisar e avaliar arquiteturas de computadores com todos os seus aspectos. Com o *Gem5*, pode-se fazer a análise de um sistema utilizando aplicações reais, como comportamento da arquitetura com Sistemas Operacionais consolidados, cargas de trabalho, dentre outras.

Além destes fatores, o simulador *Gem5* possui diversas características e estudos prévios que demonstraram a sua eficácia, acurácia e estabilidade em simulações de sistemas completos [Butko et al. 2012, Wang et al. 2013, Gutierrez et al. 2014, Souza 2015].

Com ele, é possível modelar as arquiteturas definidas e compará-las a uma arquitetura tradicional.

O conjunto de aplicações do *CAP Bench* [Souza et al. 2016] foi escolhido para a avaliação das arquiteturas. Tratam-se de aplicações voltadas para a avaliação de processadores *many-core*, como os baseados em NoCs. As aplicações do *CAP Bench* usadas neste trabalho foram desenvolvidas seguindo padrões de projetos paralelos, e estão relacionadas a seguir:

- *FAST* um algoritmo de processamento de imagens que identifica regiões de interesse em uma imagem, através do padrão paralelo *Stencil*.
- *FN* contempla a teoria dos números amigáveis que são pares de números onde um número é igual a soma dos divisores do outro, o que requer muita computação.
- *GF* também usado em processamento de imagens, é um algoritmo que aplica um filtro de suavização de imagens que desconsidera arestas em uma imagem.
- *IS* implementa um algoritmo de ordenação de inteiros baseado na estratégia *bucket-sort*.
- *KM* composto do algoritmo *K-Means*, para agrupamentos de itens através de suas similaridades.
- *LU* algoritmo de fatoração de uma matriz como o produto de uma matriz inferior (*Lower*) e uma matriz superior (*Upper*), utilizado em análise numérica para resolver sistemas de equações ou encontrar matrizes inversas.
- *TSP* consiste de um algoritmo para resolver o problema de roteamento de um caixeiro viajante hipotético. O caixeiro deve passar por N cidades, apenas uma vez por cidade, retornando para a cidade de origem pelo menor caminho possível.

A execução de cada uma das aplicações nas simulações podem mostrar resultados bastante distintos mesmo se o fluxo de dados for semelhante. Os parâmetros observados nas simulações são a largura de banda, requisições de leitura e escrita nos controladores de memória em cada roteador da NoC e o consumo de potência, também da NoC, para cada algoritmo executado.

Com a largura de banda é possível observar em quais regiões da rede há desbalanceamentos de carga. Cada roteador pode apresentar largura de banda diferente, logo, um desvio padrão alto pode indicar degradação no desempenho do sistema.

O balanceamento de acessos a memória indicam a degradação da largura de banda. O desvio padrão também em razão da média de largura de banda é utilizado para avaliar a comunicação dentro da rede-em-chip das arquiteturas simuladas. Altos valores de desvios podem indicar que há degradação na largura de banda em certos processadores.

O consumo de potência da NoC também é verificado. Isto é, o consumo dos componentes que compõem cada NoC. Desta maneira, é possível avaliar quais topologias oferecem melhores resultados em termos de consumo.

5. Resultados

Os resultados foram divididos em três seções: Análise de Distribuição de Carga, Largura de Banda e Consumo de Potência.

As arquiteturas foram nomeadas da forma X-Y-Z, onde X representa a topologia, Y representa quantos núcleos a arquitetura possui e Z representa o tamanho da memória

cache por núcleo a arquitetura possui. Como exemplo, T1-16-16 representa uma topologia *Torus* com arquitetura de 16 núcleos, 16 kB de memória *cache* por núcleo e com memórias *cache* localizadas como mostrado na Figura 1.

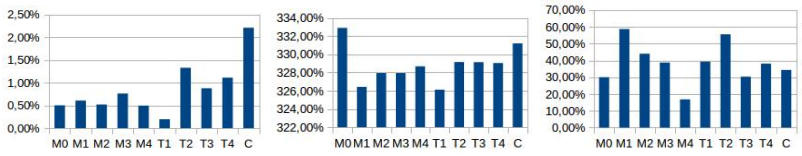
5.1. Análise de Distribuição de Cargas de Trabalho

Em relação aos balanceamento de cargas, os algoritmos FAST, IS e LU foram os escolhidos para análise, pois suas características indicam uma intensidade de comunicação alta entre os núcleos das redes [Souza et al. 2016].

Com relação ao tempo, as arquiteturas com 32 núcleos apresentaram um desempenho ligeiramente melhor do que as simulações com 16 núcleos. Em geral, as topologias organizadas em *cluster* apresentaram melhores resultados, seguida das arquiteturas T3 e T2, respectivamente, e ambas com 16 núcleos.

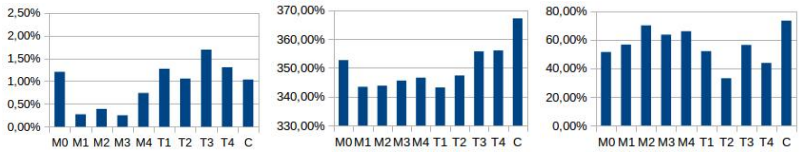
Observando o balanceamento de carga no acesso a memória, as arquiteturas mais balanceadas também foram aquelas que possuem 16 núcleos. Variando a topologia, as topologias organizadas em malhas apresentaram um balanceamento de carga maior. As arquiteturas mais balanceadas foram a T1 com 16 núcleos, M3 com 32 núcleos e a M1 com 32 núcleos, em primeiro, segundo e terceiro lugares, respectivamente.

As Figuras 2 e 3 mostram os resultados das arquiteturas comparadas a arquitetura base M0 para cada um dos três algoritmos.



(a) FAST 16 núcleos (b) IS 16 núcleos (c) LU 16 núcleos

Figura 2. Distribuição de Cargas em arquiteturas com 16 núcleos



(a) FAST 32 núcleos (b) IS 32 núcleos (c) LU 32 núcleos

Figura 3. Distribuição de Cargas em arquiteturas com 32 núcleos

A topologia *Torus* possui mais ligações do que qualquer outra topologia. A facilidade de acesso das memórias *cache* dentro da rede é maior nessa topologia. Este fator influencia na distribuição das cargas pois a distância média entre a memória e o processador é menor. A topologia em *cluster* apresenta as maiores distâncias entre os núcleos e as *caches*, mostrando resultados piores em relação as demais arquiteturas.

Por ter memórias *cache* concentradas na região central da arquitetura, a topologia *cluster* apresentou os piores resultados dentre as simulações feitas. Isso se dá pelo fato dos roteadores ligados as memórias *cache* ficarem sobrecarregados com o fluxo de dados dos diferentes núcleos daquele grupo, havendo menos caminhos onde os *bytes* possam trafegar.

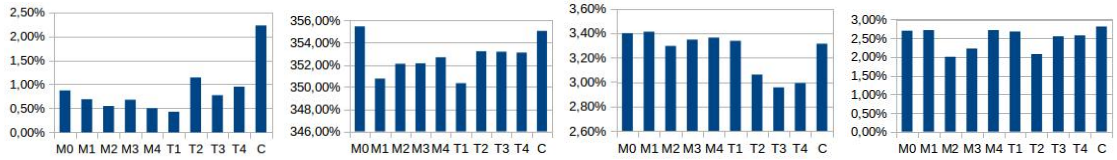
5.2. Largura de Banda

O desempenho da arquitetura em relação a largura de banda fica evidente em algoritmos que possuem um grande número de acessos a memória e um carregamento de cargas irregulares (não há padrão de acesso à memória entre os núcleos). Sendo assim, foram escolhidos os algoritmos FAST, IS, KM e TSP, em especial o KM por possuir as duas características acima e os demais apenas pelo acesso irregular a memória.

As análises mostraram que, para este quesito, a topologia escolhida influenciou bastante nos resultados entre os algoritmos. Para o algoritmo FAST e IS, a configuração T1 com 16 núcleos apresentou o melhor resultado. Já para o algoritmo KM, o melhor resultado ficou com a configuração T3. E por fim, para o algoritmo TSP, o melhor resultado foi o das configurações T2 e M2.

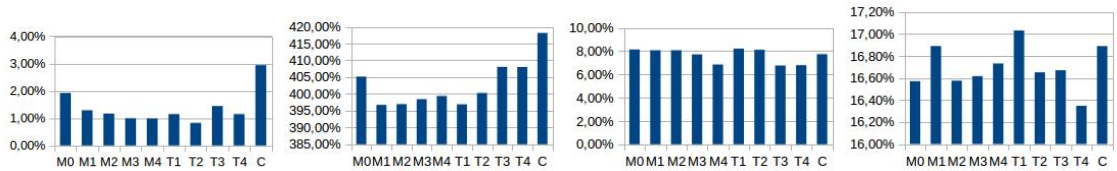
Porém, se tratando de tempo de execução, os melhores resultados foram das arquiteturas em *cluster* para esses algoritmos. Isso indica que uma arquitetura onde a largura de banda é semelhante para todos núcleos não será a que terá o melhor resultado. Na seção 5.3 mostra-se os tempos de execução de alguns destes algoritmos.

As Figuras 4 e 5 mostram os resultados das arquiteturas comparadas a arquitetura base M0 para cada um dos quatro algoritmos.



(a) FAST 16 núcleos (b) IS 16 núcleos (c) KM 16 núcleos (d) TSP 16 núcleos

Figura 4. Razões entre desvio padrão sobre média da Largura de Banda das arquiteturas com 16 núcleos



(a) FAST 32 núcleos (b) IS 32 núcleos (c) KM 32 núcleos (d) TSP 32 núcleos

Figura 5. Razões entre desvio padrão sobre média da Largura de Banda das arquiteturas com 32 núcleos

Descartando a arquitetura tradicional (M0), a que possui a maior facilidade de acesso em relação as demais são as arquiteturas organizadas em *Torus*. Novamente, as arquiteturas em *cluster* apresentaram desempenhos menores em relação as demais devido o caminho que os dados tem de percorrer para chegar nos núcleos de destino.

As arquiteturas organizadas em *cluster* também apresentaram resultados piores em relação as outras arquiteturas nesse quesito. Com os roteadores centrais, o roteamento de dados fica restrito dentro da rede a estes roteadores ligados as memórias *cache*. Sendo assim, quando a um fluxo de comunicação muito grande ao longo destes roteadores, a chance de se encaminhar os dados para processadores que estejam com cargas

maiores aumenta, pois a comunicação nos roteadores de memória está saturada. Porém, mesmo assim foram as arquiteturas que apresentaram um tempo de execução menor que as demais, mostrando que largura de banda balanceada nos roteadores da arquitetura não indicam necessariamente o melhor tempo de execução.

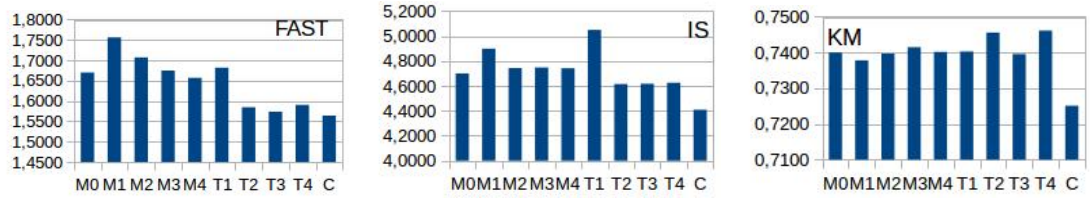
5.3. Desempenho Geral

Levando em consideração apenas o tempo de execução, duas análises foram feitas. A primeira leva em consideração a organização da arquitetura, ou seja, a topologia. A segunda leva em consideração o número de núcleos.

Com exceção dos algoritmos FN e LU, as arquiteturas organizadas em *cluster* apresentaram os melhores resultados. Isso mostra que desbalanceamento em acesso e largura de banda não necessariamente indicam uma piora no sistema, pois as arquiteturas em *cluster* não figuraram entre os melhores resultados para estes dois quesitos.

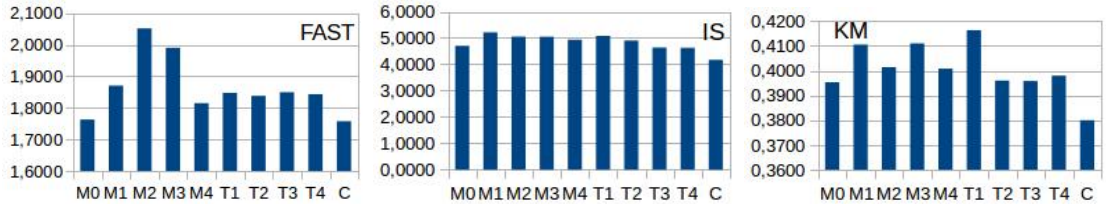
Analisando o número de núcleos, os algoritmos FAST e LU mostraram melhores resultados para configurações para 16 núcleos. Já os algoritmos FN, GF, KM e TSP mostraram resultados opostos, havendo uma melhora significativa quando se dobrava o número de núcleos presentes na arquitetura. Para o algoritmo IS, não houve influência no aumento do número de núcleos no tempo de execução do algoritmo.

As Figuras 6 e 7 mostram todos os tempos de execução dos algoritmos FAST, IS e KM em segundos.



(a) FAST (b) IS (c) KM

Figura 6. Tempos de execução em segundos dos algoritmos FAST, IS e KM para 16 núcleos



(a) FAST (b) IS (c) KM

Figura 7. Tempos de execução em segundos dos algoritmos FAST, IS e KM para 32 núcleos

5.4. Consumo de Potência e Energia

Para avaliar o consumo de cada arquitetura, todos os algoritmos foram levados em consideração. Como esperado, as arquiteturas que mostraram os maiores consumos de

potência foram aquelas com o maior número de componentes, no caso a M0 com 32 núcleos.

Em relação as topologias, as arquiteturas em malha foram as que apresentaram os menores consumos de potência. Com menos links interconectando os roteadores, são as arquiteturas mais econômicas. A topologia *cluster* possui roteadores com mais portas, por isso apresentam consumos de potência maiores, mesmo com menos ligações que a topologia em malha.

As Figuras 8 e 9 mostram o consumo de cada simulação separadas por quantidade de núcleos.

Figura 8. Consumo de potência 16 núcleos (em μW)

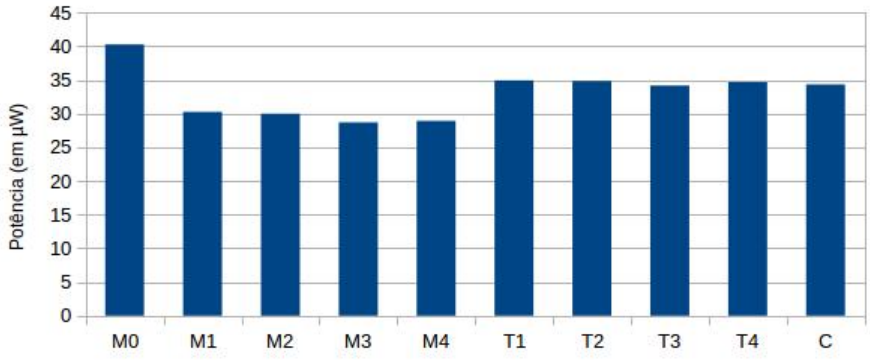


Figura 9. Consumo de potência 32 núcleos (em μW)

Observando o consumo de energia, que é o produto do consumo de potência e o tempo gasto na simulação, temos as arquiteturas com topologias *cluster* mais econômicas do que as demais, uma vez que esse tipo de topologia contempla menos interconexões, possui a maioria dos roteadores com apenas 2 portas e a comunicação centralizada em 4 e 8 roteadores para as arquiteturas de 16 e 32 núcleos de processamento, respectivamente.

As arquiteturas que apresentaram os maiores consumos de energia foram a malha com 16 memórias *cache* seguido das topologias organizadas em *Torus*. O primeiro caso se deve ao fato de que a arquitetura apresenta um maior número de componentes e no segundo, há mais interconexões entre os roteadores da rede.

Assim, a topologia *cluster* mostra-se a melhor topologia, pois apresenta os menores tempo de simulação bem como o menor custo energético dentre as arquiteturas

simuladas. As Figuras 10 e 11 mostram, em μJ, o consumo de energia das arquiteturas com a soma de todas as simulações.

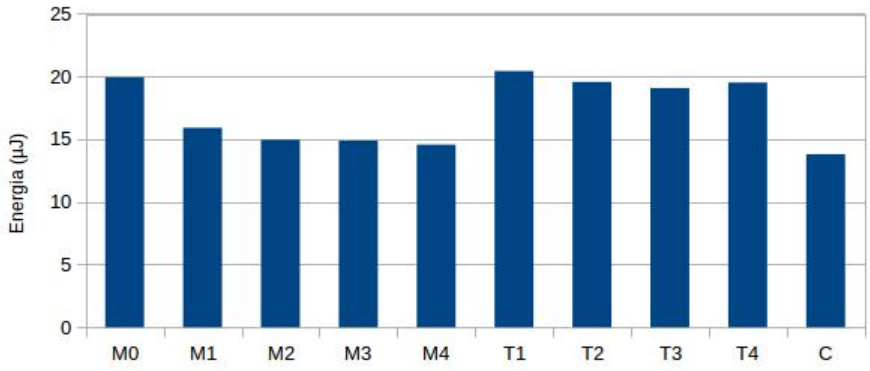


Figura 10. Consumo de energia 16 núcleos (em μJ)

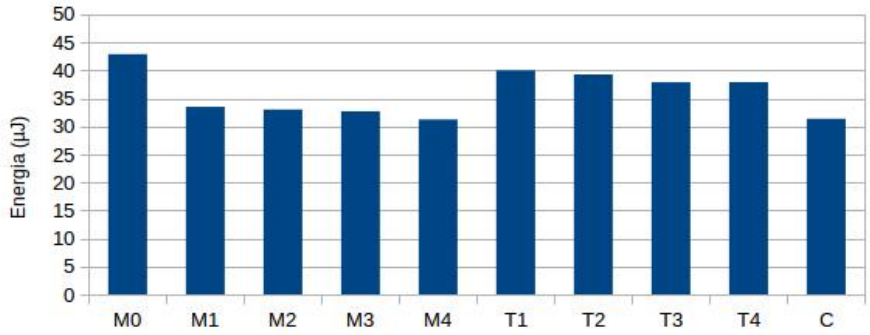


Figura 11. Consumo de energia 32 núcleos (em μJ)

6. Considerações finais

Este trabalho apresentou uma avaliação de Redes-em-Chip simuladas com o *Gem5*, buscando analisar o balanceamento de carga, largura de banda e o consumo de potência de cada arquitetura simulada. Variando o tipo de topologia entre Malha, *Torus* e *Cluster*, e o número de *Clusters* por arquitetura, foram calculados os valores de desvio padrão em relação a média de cada uma das métricas para mostrar balanceamento de algoritmos do CAP Bench nas arquiteturas simuladas.

Os resultados mostraram um melhor desempenho geral da topologia *cluster* em relação as demais topologias. Balanceamento de Carga e Disponibilidade de Recursos são características diretamente influenciadas pela facilidade de comunicação dentro da rede. Porém, os resultados mostram que a escolha da topologia influência mais o desempenho do sistema do que o balanceamento de carga e a largura de banda entre os núcleos de uma rede em chip.

Todas as simulações foram feitas com *caches* de mesmo tamanho. Como trabalhos futuros, variar este parâmetro dentro das simulações pode mostrar qual sua influência dentro do sistema. Apenas o balanceamento de carga e a largura de banda dos processadores não bastam para determinar se há degradação do desempenho sistema pela comunicação. Se analisados em conjunto com outros fatores, como picos de processamento, ociosidade e

latência, será possível indicar se a arquitetura possui gargalos específicos de comunicação na rede-em-chip.

7. Agradecimento

Os autores agradecem ao CNPq, CAPES e FAPEMIG pelo suporte no desenvolvimento do trabalho.

Referências

Asanovic, K., Bodik, R., Demmel, J., Keaveny, T., Keutzer, K., Kubiatowicz, J., Morgan, N., Patterson, D., Sen, K., Wawrzynek, J., et al. (2009). A view of the parallel computing landscape. *Communications of the ACM*, 52(10):56–67.

Benini, L. and Micheli, G. D. (2002). Networks on chips: a new soc paradigm. *Computer*, 35(1):70–78.

Binkert, N., Beckmann, B., Black, G., Reinhardt, S. K., Saidi, A., Basu, A., Hestness, J., Hower, D. R., Krishna, T., Sardashti, S., Sen, R., Sewell, K., Shoaib, M., Vaish, N., Hill, M. D., and Wood, D. A. (2011). The gem5 simulator. *ACM SIGARCH Computer Architecture News*, 39(2):1–7.

Butko, A., Garibotti, R., Ost, L., and Sassatelli, G. (2012). Accuracy evaluation of gem5 simulator system. In *Reconfigurable Communication-centric Systems-on-Chip (ReCoSoC), 2012 7th International Workshop on*, pages 1–7. IEEE.

Camacho, J., Flich, J., Duato, J., Eberle, H., and Olesinski, W. (2011). A power-efficient network on-chip topology. In *Proceedings of the Fifth International Workshop on Interconnection Network Architecture: On-Chip, Multi-Chip*, pages 23–26. ACM.

Freitas, H. C., Santos, T. G. S., and Navaux, P. O. A. (2008). Design of programmable noc router architecture on fpga for multi-cluster nocs. *Electronics Letters*, 44(16):969–971.

Freitas, H. C. d., Alves, M. A. Z., and Navaux, P. O. A. (2009). Noc e nuca: Conceitos e tendências para arquiteturas de processadores many core. In *IX Escola Regional de Alto Desempenho (ERAD)*, pages 364–366, Caxias do Sul. SBC.

Gutierrez, A., Pusdesris, J., Dreslinski, R. G., Mudge, T., Sudanthi, C., Emmons, C. D., Hayenga, M., and Paver, N. (2014). Sources of error in full-system simulation. In *Performance Analysis of Systems and Software (ISPASS), 2014 IEEE International Symposium on*, pages 13–22. IEEE.

Ho, R., Mai, K. W., and Horowitz, M. A. (2001). The future of wires. *Proceedings of the IEEE*, 89(4):490–504.

Shalf, J., Dosanjh, S., and Morrison, J. (2010). Exascale computing technology challenges. In *High Performance Computing for Computational Science (VECPAR)*, pages 1–25. Springer, Berkeley, USA.

Simon, H. (2012). Barriers to exascale computing. In *High Performance Computing for Computational Science (VECPAR)*, pages 1–3. Springer, Kope, Japan.

Souza, M. A. (2015). Exploração de espaço de projeto de arquiteturas de processadores many-core baseados em redes-em-chip com uso de simulação de sistemas completos. Master's thesis, Pontifícia Univarsidade Católica de Minas Gerais, Brazil.

Souza, M. A., Penna, P. H., Queiroz, M. M., Pereira, A. D., Góes, L. F. W., Freitas, H. C., Castro, M., Navaux, P. O., and Méhaut, J.-F. (2016). Cap bench: a benchmark suite for performance and energy evaluation of low-power many-core processors. *Concurrency and Computation: Practice and Experience*, pages n/a–n/a. cpe.3892.

Udipi, A. N., Muralimanohar, N., and Balasubramonian, R. (2010). Towards scalable, energy-efficient, bus-based on-chip networks. In *HPCA-16 2010 The Sixteenth International Symposium on High-Performance Computer Architecture*, pages 1–12, Bangalore. IEEE, IEEE.

Wang, R., Chen, L., and Pinkston, T. M. (2013). An analytical performance model for partitioning off-chip memory bandwidth. In *Parallel & Distributed Processing (IPDPS), 2013 IEEE 27th International Symposium on*, pages 165–176. IEEE.

Xu, T. C., Schley, G., Liljeberg, P., Radetzki, M., Plosila, J., and Tenhunen, H. (2013). Optimal placement of vertical connections in 3d network-on-chip. *Journal of Systems Architecture*, 59(7):441–454.