Memória *Scratchpad* Compartilhada para Comunicação Eficiente em Arquiteturas *Multi-Core*[*]

Francis Birck Moreira, Eduardo Henrique Molina da Cruz, Marco Antonio Zanata Alves, Philippe Olivier Alexandre Navaux

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

{fbmoreira, ehmcruz, mazalves, navaux}@inf.ufrgs.br

Abstract. *Scratchpad memories are simple arrays of memory used to improve access to data. As a regularly used memory within single-purpose processors due to lower energy requirements, such as mobile phones and other embedded processors, a scratchpad memory can be seen as a viable choice for general purpose systems if it can compete with the current cache memory systems in usage.*
This paper shows the evaluation of a scratchpad memory, presenting results that show the potential gain that could be achieved by using scratchpad memories in combination with cache memories.

Resumo. *Memórias* scratchpad *são simples vetores de memória usados para melhorar o acesso a dados. Sendo regularmente usada em processadores de propósito específico devido ao seu consumo menor de energia, tal como telefones móveis e outros processadores embarcados, uma memória* scratchpad *pode ser uma escolha viável para sistemas de propósito genérico se puder competir com os sistemas atuais de memórias* cache *em uso.*
Este artigo mostra a avaliação de uma memória scratchpad*, apresentando resultados que demonstram o ganho potencial a ser alcançado no uso de memórias* scratchpad *em combinação com memórias* cache*.*

1. Introdução

As arquiteturas *multi-core* estão na vanguarda do desenvolvimento de arquiteturas de alto desempenho. Em tais arquiteturas, a memória representa um gargalo em potencial, já que, quanto maior a quantidade de núcleos de processamento, maior a vazão de dados requerida. Para suprir tal necessidade, memórias caches são empregadas de maneira a mascarar a latência da memória primária, sendo que vários níveis de hierarquia são utilizados, podendo até mesmo ser compartilhados por mais de um núcleo. Entretanto, a presença de dados compartilhados por diferentes memórias caches pode acarretar em perda de desempenho, tanto por sobrecargas impostas por protocolos de coerência quanto por pior utilização da cache devido à duplicação de dados.

Memórias *Scratchpad* (SPM - *ScratchPad Memories*) estão presentes em muitos sistemas embarcados. Trata-se de uma memória especializada, no qual é responsabilidade do programador, compilador ou sistema operacional selecionar quais dados serão

[*]Trabalho parcialmente apoiado pelo CNPq e CAPES.

armazenados. As pesquisas relacionadas ao uso de memórias *scratchpad* em arquiteturas *multi-core* mais recentes apresentam ganhos principalmente em termos de consumo de energia e área de ocupação em relação à cache tradicional, já que o uso de comparadores e *tags* é desnecessário. Também pode-se obter ganhos de desempenho ao empregar o *scratchpad* [Suhendra et al. 2010], já que possui uma latência inferior a uma memória *cache* de mesmo tamanho.

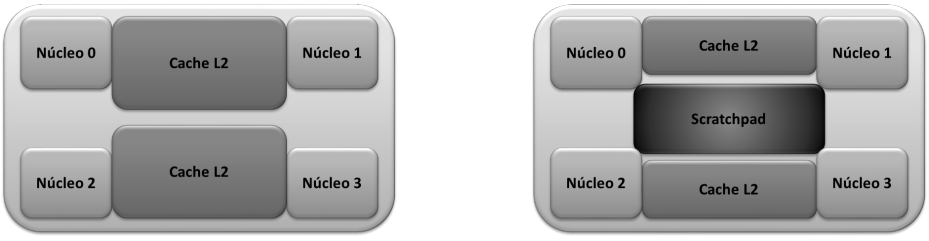
A utilização de memórias *scratchpad* em arquiteturas *multi-core* é discutida em [Yanamandra et al. 2008]. Ao integrar-se o *scratchpad* à hierarquia de memoria cache existente, uma escolha eficaz do conteúdo do *scratchpad* poderia minimizar problemas gerados por compartilhamento de dados, revelando um potencial ganho em termos de energia e desempenho. Entretanto, restrições e problemas de retrocompatibilidade surgem ao inserir o *scratchpad* nos sistemas atuais, já que seu uso é estritamente dependente do programa em execução, o que obrigaria uma re-estruturação do *software* existente.

Em [Nguyen et al. 2009], é demonstrada uma técnica para resolução do problema de retrocompatibilidade ao criar-se um instalador de *software* customizado, o qual decide a alocação da memória *scratchpad* exatamente antes da primeira execução do programa. Obtêm-se o tamanho da memória *scratchpad* e o instalador também modifica o executável de acordo com a alocação escolhida. Em [Francesco et al. 2004], é avaliada a implementação de *hardware* adicional na forma de um DMA(*Direct Memory Access*) como solução para o problema de alocação em tempo de execução da memória *scratchpad*, permitindo programação de nível mais abstrato e um mecanismo integrado para alocação do *scratchpad* durante a execução.

Este trabalho avalia o desempenho de *scratchpads* em arquiteturas *multi-core*. O simulador Simics [Magnusson et al. 2002] foi utilizado como plataforma de testes, sendo instrumentado de maneira a simular um *scratchpad* em meio à hierarquia de memória. O artigo está organizado da seguinte maneira: a Seção 2 apresenta os resultados dos experimentos realizados e a Seção 3 contém as conclusões e trabalhos futuros.

2. Experimentos com a Memória *Scratchpad*

Para analisar o ganho do uso do *scratchpad* em uma arquitetura *multi-core* atual, foi modelado no simulador Simics um processador com 4 núcleos de arquitetura UltraSPARC II executando a 3.0 GHz, com a hierarquia de memória conforme apresentado na Figura 1. Os parâmetros arquiteturais foram baseados no processador Quad-Core da Intel de arquitetura *Harpertown*, sendo que as latências de acesso foram obtidas com auxílio da ferramenta CACTI versão 6.5 [Muralimanohart et al. 2007].



(a) Arquitetura Base. (b) Arquitetura com SPM.

Figura 1. Ilustração das arquiteturas modeladas.

Conf.	L1			L2			Scratchpad	
	Tamanho da Cache	Conjunto Assoc.	Latência (Ciclos)	Tamanho da Cache	Conjunto Assoc.	Latência (Ciclos)	Tamanho do SPM	Latência (Ciclos)
1	32+32 KB	8 vias	3	6+6 MB	24 vias	12	-	-
2	32+32 KB	8 vias	3	6+6 MB	24 vias	12	8 MB	2
3	32+32 KB	8 vias	3	1+1 MB	24 vias	8	8 MB	10
4	32+32 KB	8 vias	3	1+1 MB	24 vias	8	8 MB	2

Tabela 1. Configurações utilizadas nos experimentos.

A Tabela 1 contém as configurações utilizadas nos experimentos. As configurações foram modeladas de forma a demonstrar o efeito da presença de um *scratchpad* na hierarquia de memória com *caches*, e a diferença entre o ganho máximo alcançável teoricamente e o máximo alcançável com uma memória *scratchpad* de tamanho suficiente para suportar quantidades grandes de dados.

Como carga de trabalho, fora utilizado o *benchmark* IS (*Integer Sort*) que faz parte do NPB (*NAS Parallel Benchmarks*) com o tamanho pré-definido W dos dados de entrada, que é o mais indicado para simulações. Essa carga de trabalho é paralelizada com OpenMP, sendo que nos testes feitos foi alocada apenas uma *thread* por núcleo.

Os resultados em termos de tempo de execução e transações nos diversos níveis de memória estão presentes na Figura 2.

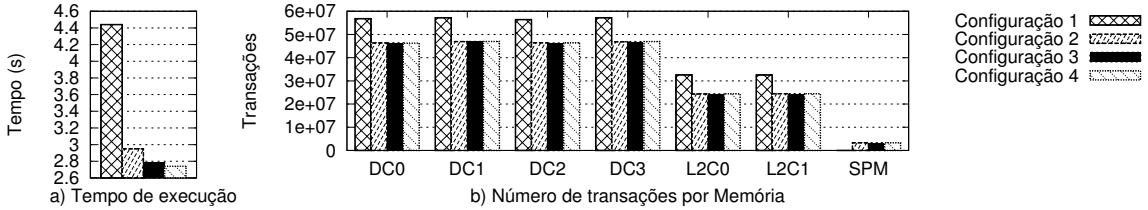


Figura 2. Resultados obtidos no benchmark IS.W.

O ganho de 33,55% observado na segunda configuração representa uma alocação perfeita de 3 das 4 estruturas principais do programa no *scratchpad*, o que seria pouco realístico, pois haveria apenas 16 KB para alocar estruturas de 4,25 MB, gerando constantes trocas dos dados alocados no SPM, acarretando latências adicionais.

Porém, na terceira configuração, observam-se ganhos superiores (37,38%). A memória *scratchpad* tem tamanho razoável, alocando as mesmas 3 estruturas da segunda configuração (4,25 MB). De fato, a memória *scratchpad* poderia conter estruturas de múltiplos processos caso fosse feita uma seleção mais precisa da estrutura de dados mais crítica de cada processo.

A quarta configuração, com ganhos de 38,28%, demonstra o motivo do ganho de desempenho da terceira configuração em relação à segunda. Observando que o ganho de desempenho não se deve unicamente à baixa latência da memória *scratchpad*, mas sim da redução de transações das memórias *cache* e da redução da latência da memória *cache* L2 e sua taxa de erros.

A redução do número de transações nos diversos níveis de memória *cache*, somada à garantia de que as estruturas de dados compartilhadas mais requisitadas pela aplicação

paralela estarão no *scratchpad* e serão acessadas com menor latência, representam os principais motivos de ganho de desempenho nas arquiteturas que utilizaram a memória *scratchpad*.

3. Conclusões e Trabalhos Futuros

As memórias *scratchpad* apresentam características favoráveis para sistemas com requisitos de área, energia e latência. Por isto, as memórias *scratchpad* representam uma solução para o compartilhamento de dados entre vários núcleos em arquiteturas *multi-core*, uma vez que diversos programas paralelos compartilham dados e requerem garantia de rápido acesso aos dado compartilhados.

Neste trabalho, foi avaliado o uso de uma memória *scratchpad* compartilhada por vários núcleos de processamento em um sistema de propósito geral. A execução da carga de trabalho IS.W do *benchmark* NPB apontou ganhos no desempenho de aproximadamente 37% ao reduzir o tamanho do segundo nível de memórias *cache* de 12 MB para 2 MB e adicionar uma memória *scratchpad* de 8 MB no sistema.

Como trabalhos futuros, pretende-se estudar maneiras de gerenciar os dados alocados no *scratchpad* de forma a permitir a total integração desta memória às atuais arquiteturas *multi-core*. Além disso, serão pesquisados métodos para adicionar marcações no código de forma a solicitar o uso do *scratchpad* ao sistema operacional.

Referências

Francesco, P., Marchal, P., Atienza, D., Benini, L., Catthoor, F., and Mendias, J. M. (2004). An integrated hardware/software approach for run-time scratchpad management. In *Proceedings of the 41st annual Design Automation Conference*, DAC '04, pages 238–243, New York, NY, USA. ACM.

Magnusson, P. S., Christensson, M., Eskilson, J., Forsgren, D., Hallberg, G., Hogberg, J., Larsson, F., Moestedt, A., and Werner, B. (2002). Simics: A full system simulation platform. *Computer*, 35:50–58.

Muralimanohart, N., Balasubramonian, R., and Jouppi, N. (2007). Optimizing nuca organizations and wiring alternatives for large caches with cacti 6.0. In *40th Annual IEEE/ACM International Symposium on Microarchitecture.*, MICRO '04, pages 3 –14.

Nguyen, N., Dominguez, A., and Barua, R. (2009). Memory allocation for embedded systems with a compile-time-unknown scratch-pad size. *ACM Trans. Embed. Comput. Syst.*, 8:21:1–21:32.

Suhendra, V., Roychoudhury, A., and Mitra, T. (2010). Scratchpad allocation for concurrent embedded software. *ACM Trans. Program. Lang. Syst.*, 32:13:1–13:47.

Yanamandra, A., Cover, B., Raghavan, P., Irwin, M., and Kandemir, M. (2008). Evaluating the role of scratchpad memories in chip multiprocessors for sparse matrix computations. In *IEEE International Symposium on Parallel and Distributed Processing.*, IPDPS '08, pages 1 –10.