Online Thread and Data Mapping Using the Memory Management Unit

Eduardo H. M. Cruz, Philippe O. A. Navaux

[1] Instituto de Informática – Universidade Federal do Rio Grande do Sul (UFRGS)

{ehmcruz, navaux}@inf.ufrgs.br

Resumo. *As arquiteturas de computadores atuais incluem complexas hierarquias de memória que introduzem diferentes tempos de acesso à memória. Uma das soluções adotadas para reduzir o tempo de acesso é aumentar a localidade dos acessos à memória através do mapeamento de threads e dados. Nesta tese de doutorado, são propostas soluções inovadoras para identificar um mapeamento que otimize o acesso à memória fazendo uso da unidade de gerência de memória para monitor os acessos. Na avaliação experimental, as soluções melhoraram o desempenho em até 39% e a eficiência energética em até 12,2%. Isto se deu por uma redução substancial da quantidade de faltas na cache, tráfego entre processadores e acessos à bancos de memória remotos.*

1. Introdução

O paralelismo a nível de *threads* tem aumentado em arquiteturas modernas devido ao maior número de núcleos por processador e processadores por sistema. Devido a isto, a complexidade das hierarquias de memória também aumenta. Tais hierarquias podem incluir vários níveis de *cache* privadas ou compartilhadas, bem como tempos de acesso não uniformes à bancos de memória. Um desafio importante para essas arquiteturas é a movimentação de dados entre os núcleos, *caches* e bancos de memória, que ocorre quando um núcleo realiza uma operação de memória. Neste contexto, a redução da movimentação de dados é uma meta importante para arquiteturas futuras para aumentar o desempenho e diminuir o consumo de energia [Borkar and Chien 2011]. Uma das soluções para reduzir a movimentação de dados é melhorar a localidade do acesso à memória através do mapeamento de *threads* e dados [Feliu et al. 2012].

Mecanismos de mapeamento baseados no compartilhamento de dados visam aumentar a localidade mantendo *threads* que compartilham um grande volume de dados próximas na hierarquia de memória (*mapeamento de threads*), e mantendo os dados que as *threads* acessam próximos destas *threads* (*mapeamento de dados*). O desempenho e eficiência energética são aumentados por três razões principais. Em primeiro lugar, as faltas de *cache* são reduzidas, pois diminui o número de invalidações de linha de *cache* em dados compartilhados e reduz a duplicação de linhas de *cache* em várias *caches*. Em segundo lugar, a localidade dos acessos à memória é aumentada através do mapeamento de dados para o nó onde são mais acessados. Em terceiro lugar, a utilização das interconexões no sistema é melhorada pela redução do tráfego em interconexões lentas entre os processadores, utilizando interconexões dentro dos processadores, mais eficientes. A

Este trabalho foi parcialmente apoiado pelo Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq), pelo programa EU H2020 MCTI/RNP-Brasil projeto HPC4E n. 689772, e pela Intel.

falha em identificar o padrão de acesso à memória de uma aplicação pode levar a mapeamentos ruins que reduzem o desempenho e a eficiência energética.

O desenvolvimento de técnicas de mapeamento para ambientes de memória compartilhada é desafiador. Primeiramente, é necessário coletar informações sobre o padrão de compartilhamento de dados das aplicações. Entretanto, a comunicação usando memória compartilhada é implícita e ocorre através do acesso a uma mesma região de memória por diferentes *threads*. Quando o mapeamento é realizado de forma *online*, isto é, durante a execução das aplicações, esta coleta de informações apresenta uma dificuldade adicional, pois o *hardware* existente não provê mecanismos para identificar quais *threads* acessam quais dados. Outro ponto importante é a elaboração de um mecanismo que consiga usar as informações sobre o acesso à memória de forma a se ganhar desempenho e diminuir o consumo energético. Esse mecanismo tem que ser capaz de identificar padrões de acesso e realizar o mapeamento das *threads* e dados de maneira inteligente, evitando sobrecargas. Logo, as duas questões são: *(1) como detectar quais threads acessam quais dados de forma online?* e *(2) como usar estas informações para mapear as threads e dados, também de forma online, melhorando o desempenho e a eficiência energética?*

Fazendo uma análise do estado-da-arte, a maioria dos trabalhos realiza mapeamento de *threads* ou dados apenas, mas não os dois juntos, não sendo capazes de otimizar adequadamente as aplicações. A maioria dos mecanismos que realizam ambos os mapeamentos têm várias desvantagens. Mecanismos estáticos [Cruz et al. 2011] dependem da informação de execuções anteriores, sendo limitados a aplicações cujo comportamento não mudam entre execuções ou durante a execução. Outros mecanismos são *online* [Broquedis et al. 2010], mas exigem anotações de código fonte por parte do programador para funcionar. Alguns [Dashti et al. 2013] usam amostragem e tem uma alta sobrecarga quando aumentamos a quantidade de amostras para alcançar maior precisão. Outros mecanismos dependem de estatísticas indiretas obtidas por contadores de *hardware*, que não representam com precisão o comportamento dos acessos à memória de aplicações paralelas. Várias propostas exigem arquiteturas, APIs ou linguagens de programação específicas, o que limitam suas aplicabilidades. Analisando o estado-da-arte, podemos concluir que, anteriormente a esta tese, *não existia um mecanismo online que podia ser aplicado a qualquer aplicação baseada em memória compartilhada, em que aumentar a sua precisão não aumentasse drasticamente a sua sobrecarga*.

Nesta tese de doutorado, foram propostos mecanismos que preenchiam esta lacuna do estado-da-arte, para realizar o mapeamento de *threads* e dados a fim de se melhorar a localidade dos acessos à memória. As propostas realizam o mapeamento tanto de *threads* como de dados de forma *online*, otimizando o uso de *cache* e diminuindo acessos remotos e uso de interconexões entre processadores. Elas são implementadas diretamente na unidade de gerência de memória dos processadores (MMU – *memory management unit*), o que permite monitorar muito mais acessos à memória do que os trabalhos relacionados de uma forma não intrusiva. Desta forma, pode-se conseguir uma precisão muito maior nos padrões detectados, mantendo uma baixa sobrecarga. Isto acontece porque os mecanismos aqui propostos são híbridos, tendo um suporte em *hardware*, que permite um monitoramento dos acessos à memória de forma transparente. O sistema operacional utiliza as informações geradas pelos mecanismos e realiza o mapeamento de *threads* e dados de forma transparente ao usuário, aumentando o desempenho e a eficiência energética.

Este resumo está organizado da seguinte forma. A Seção 2 explica os mecanismos propostos nesta tese. A Seção 3 descreve os experimentos. A Seção 4 expõe as conclusões. A Seção 5 contém as publicações relacionadas à tese.

2. Mecanismos Propostos para Mapeamento Baseado em Compartilhamento

Na tese, foram propostos 3 mecanismos para detectar o padrão de acesso à memória para o mapeamento. Os mecanismos foram implementados na MMU, fazendo uso do artifício de memória virtual. Os mecanismos propostos são chamados **LAPT**, **SAMMU** e **IPM**. As principais diferenças entre eles consistem em quais informações são coletadas. O LAPT usa faltas na translation lookaside buffer (TLB), o SAMMU inclui um contador de acessos por entrada da TLB, e o IPM usa o tempo que cada página tem sua entrada na TLB. O LAPT possui menor precisão e sobrecarga, o SAMMU maior precisão e sobrecarga, enquanto o IPM possui a melhor relação entre precisão e sobrecarga. Neste resumo, é dado uma breve explicação sobre o IPM. Os outros mecanismos não serão tratados devido a restrições de espaço do artigo.

Uma visão geral de alto nível do funcionamento do MMU, TLB e do IPM é ilustrado na Figura 1. Em cada falta na TLB, o IPM armazena, na memória principal, o *time stamp* do momento que ocorreu a falta. Se uma entrada da TLB deve ser evitada para armazenar a nova entrada, o IPM carrega o *time stamp* correspondente à entrada TLB evitada, salvo na etapa anterior. Subtraindo-se o *time stamp* carregado do *time stamp* do momento da evicção, o IPM sabe quanto tempo a entrada ficou armazenada na TLB. O IPM usa essa diferença de tempo para atualizar as estatísticas da página. Por fim, o IPM notifica o sistema operacional se uma migração de página pode melhorar o desempenho.

Ao conhecimento dos autores, as propostas desta tese foram os primeiros mecanismos a adicionar à MMU a capacidade de monitorar os acessos à memória e realizar os mapeamentos de dados e *threads* via sistema operacional. Além disso, o SAMMU e o IPM foram os primeiros mecanismos a detectar o padrão de acesso à memória para o mapeamento completamente a nível de *hardware*, considerando praticamente todos os acessos à memória para atingir uma alta precisão, ao mesmo tempo que mantém uma baixa sobrecarga. Na tese, como uma contribuição secundária, é proposto também um algoritmo de mapeamento de *threads* chamado **EagerMap**, que atinge uma alta eficiência por se basear em características comuns à padrões de acesso à memória.

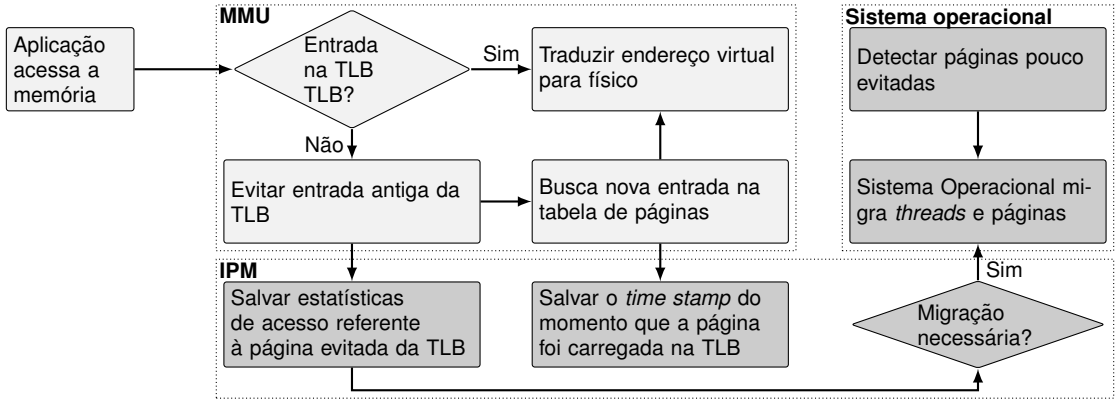

Figura 1. Visão geral da MMU, do IPM e sistema operacional.

3. Resultados Obtidos

Foram realizados experimentos em um simulador completo de sistema, uma máquina real com TLB gerida via *software* e duas máquinas reais com TLB gerida via *hardware*. Como as propostas desta tese são extensões para o *hardware* atual, nas máquinas reais com TLB gerida via *hardware*, foram usadas uma ferramenta de instrumentação de binários para gerar informações sobre os mapeamentos de *threads* e de dados. No simulador e na máquina real com TLB gerida via *software*, as propostas foram implementadas nativamente. Como carga de trabalho, foram utilizados 23 aplicações de *benchmarks* conhecidos (NAS e PARSEC) e uma aplicação real de simulação sísmica (Ondes3d). Essa ampla variedade de arquiteturas e carga de trabalho proveu resultados sólidos que provam a aplicabilidade e eficiência das propostas.

Como principais resultados, o tempo de execução das aplicações foi reduzido em até 39% e a eficiência energética foi melhorada em até 12,2%. Isto foi possível graças a uma expressiva diminuição da quantidade de faltas na *cache*, acessos à bancos de memória remotos, e redução de tráfego nas interconexões. A análise dos resultados permitiu analisar de forma precisa as características de acesso à memória das aplicações que podem se beneficiar do mapeamento. Em comparação a outras políticas de mapeamento do estado-da-arte, as propostas desta tese apresentaram melhores resultados para a maioria das aplicações. Além disso, as propostas apresentaram resultados semelhantes ao melhor mapeamento teoricamente possível, demonstrando suas eficácias.

4. Conclusão

Nesta tese de doutorado, foram propostas novas soluções para aumentar desempenho utilizando o mapeamento de *threads* e dados. As soluções suprem uma lacuna do estado-da-arte: são capazes de atingir alta precisão com uma sobrecarga baixíssima, de forma *online* e transparente. Para tal, as soluções fazem uso da MMU presente nos processadores, pois a MMU de cada núcleo tem acesso a informações sobre todos os acessos à memória. Com o suporte de *hardware* proposto, os mecanismos aqui apresentados são capazes de oferecer uma alta precisão com uma baixa sobrecarga de desempenho. Além disso, foi proposto um algoritmo de mapeamento de *threads* que gera mapeamentos de alta qualidade, porém com uma sobrecarga muito mais baixa que o estado-da-arte.

Os mecanismos foram avaliados em três ambientes diferentes: um simulador de sistema completo, uma máquina real com TLB gerida por *software* e duas máquinas reais com TLB gerida por *hardware*. Os mecanismos proveram melhorias substanciais em todos os ambientes, reduzido o tempo de execução em até 39% e melhorando a eficiência energética em até 12,2%. Os resultados obtidos na tese foram melhores que os do estado-da-arte. O custo de implementação em *hardware* é extremamente baixo, representando menos de 0,02% de área de circuito adicional em um processador moderno considerando-se o mecanismo proposto IPM, que possui a melhor relação de custo$\times$benefício entre as propostas.

Pode-se concluir da tese algo muito importante: fabricantes de processadores poderiam aumentar substancialmente o desempenho e eficiência energética de seus sistemas ao integrar no circuito módulos que monitorassem o padrão de acesso à memória, permitindo assim que o sistema operacional realizasse o mapeamento de *threads* e dados usando tais informações de forma *online* e transparente para o usuário.

5. Publicações em Conferências e Periódicos

Diversos artigos foram publicados durante o doutorado em conferências e periódicos de grande importância para as áreas de arquitetura de computadores, sistemas operacionais e processamento paralelo. Os periódicos incluem CSUR, TACO, PARCO, JPDC, CCPE, TPDS, PEVA e IJHPCA. As conferências incluem Euro-Par, CCGRID, PDP, SBAC-PAD, PACT e IPDPS. O artigo publicado no TACO foi também convidado para apresentação na conferência HiPEAC 2017. Em uma das submissões, para a conferência PDP 2015, o artigo recebeu o prêmio de **melhor artigo da conferência**. A ampla variedade de conferências e periódicos de qualidade que aceitaram os trabalhos demonstram a importância da tese e seus trabalhos derivados.

A seguir, a lista de todos os artigos publicados em temas relacionados à tese de doutorado. O Qualis das conferências e periódicos foram incluídos para mostrar a relevância dos mesmos. É importante mencionar que o periódico TACO ainda não foi incluído no Qualis, entretanto, como é um periódico de alta qualidade e seu *h5-index* é 23, ele seria classificado com o Qualis A1 ou A2.

1. Diener, Cruz, Alves, Navaux, Koren. *Affinity-Based Thread and Data Mapping in Shared Memory Systems.* ACM Computing Surveys (CSUR), Volume 49, Issue 4, 2017. **Qualis A1**.
2. Cruz, Diener, Pilla, Navaux. *Hardware-assisted thread and data mapping in hierarchical multicore architectures.* ACM Transactions on Architecture and Code Optimization (TACO) 13, 3, 2016. **h5-index 23 (Qualis aproximado A1 ou A2)**.
3. Cruz, Diener, Pilla, Navaux. *A Sharing-Aware Memory Management Unit for Online Mapping in Multi-Core Architectures*. Euro-Par, 2016. **Qualis A1**.
4. Cruz, Diener, Alves, Pilla, Navaux. *LAPT: A Locality-Aware Page Table for thread and data mapping*. Parallel Computing (PARCO), 2016. **Qualis A2**.
5. Carreño, Diener, Cruz, Navaux. *Automatic Communication Optimization of Parallel Applications in Public Clouds*. IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID), 2016. **Qualis A1**.
6. Diener, Cruz, Alves, Navaux. *Communication in Shared Memory: Concepts, Definitions, and Efficient Detection*. Euromicro International Conference on Parallel, Distributed, and Network-Based Processing (PDP), 2016. **Qualis A2**.
7. Diener, Cruz, Navaux. *Modeling memory access behavior for data mapping*. International Journal of High Performance Computing Applications (IJHPCA), 2016. **Qualis B1**.
8. Cruz, Diener, Navaux. *Communication-Aware Thread Mapping Using the Translation Lookaside Buffer*. Concurrency and Computation: Practice and Experience (CCPE), v. 22, n. 6, 2015. **Qualis A2**.
9. Cruz, Diener, Pilla, Navaux. *An Efficient Algorithm for Communication-Based Task Mapping*. Euromicro International Conference on Parallel, Distributed, and Network-Based Processing (PDP), 2015. **(Prêmio de melhor artigo)**. **Qualis A2**.
10. Diener, Cruz, Alves, Navaux, Busse, Heiss. *Kernel-Based Thread and Data Mapping for Improved Memory Affinity*. IEEE Transactions on Parallel and Distributed Systems (TPDS), v. 27, 2015. **Qualis A1**.
11. Diener, Cruz, Alves, Alhakeem, Navaux, Heiss. *Locality and Balance for Communication-Aware Thread Mapping in Multicore Systems*. Euro-Par, 2015. **Qualis A1**.

12. Diener, Cruz, Pilla, Dupros, Navaux. *Characterizing Communication and Page Usage of Parallel Applications for Thread and Data Mapping*. Performance Evaluation (PEVA), 2015. **Qualis A2**.
13. Diener, Cruz, Navaux, Busse, Heiss. *Communication-Aware Process and Thread Mapping Using Online Communication Detection*. Parallel Computing (PARCO), 2015. **Qualis A2**.
14. Diener, Cruz, Navaux. *Locality vs. Balance: Exploring Data Mapping Policies on NUMA Systems*. Euromicro International Conference on Parallel, Distributed, and Network-Based Processing (PDP), 2015. **Qualis A2**.
15. Cruz, Diener, Alves, Navaux. *Dynamic thread mapping of shared memory applications by exploiting cache coherence protocols*. Journal of Parallel and Distributed Computing (JPDC), v. 74, n. 3, 2014. **Qualis A2**.
16. Cruz, Diener, Alves, Pilla, Navaux. *Optimizing Memory Locality Using a Locality-Aware Page Table*. International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD), 2014. **Qualis B1**.
17. Diener, Cruz, Navaux, Busse, Heiss. *kMAF: Automatic Kernel-Level Management of Thread and Data Affinity*. International Conference on Parallel Architectures and Compilation Techniques (PACT), August 2014. **Qualis A2**.
18. Diener, Cruz, Navaux. *Communication-Based Mapping using Shared Pages*. IEEE International Parallel and Distributed Processing Symposium (IPDPS), 2013. **Qualis A1**.
19. Cruz, Diener, Navaux. *Using the Translation Lookaside Buffer to Map Threads in Parallel Applications Based on Shared Memory*. IEEE International Parallel and Distributed Processing Symposium (IPDPS), 2012. **Qualis A1**.

Há ainda dois artigos submetidos, um para o periódico **ACM Transactions on Computer Systems (TOCS)**, e outro para o periódico **ACM Transactions on Parallel Computing (TOPC)**.

Referências

[Borkar and Chien 2011] Borkar, S. and Chien, A. A. (2011). The Future of Microprocessors. *Communications of the ACM*, 54(5):67–77.

[Broquedis et al. 2010] Broquedis, F., Aumage, O., Goglin, B., Thibault, S., Wacrenier, P.-A., and Namyst, R. (2010). Structuring the execution of OpenMP applications for multicore architectures. In *IEEE International Parallel & Distributed Processing Symposium (IPDPS)*, pages 1–10.

[Cruz et al. 2011] Cruz, E., Alves, M., Carissimi, A., Navaux, P., Ribeiro, C., and Mehaut, J. (2011). Using memory access traces to map threads and data on hierarchical multi-core platforms. In *IEEE International Symposium on Parallel and Distributed Processing Workshops and Phd Forum (IPDPSW)*.

[Dashti et al. 2013] Dashti, M., Fedorova, A., Funston, J., Gaud, F., Lachaize, R., Lepers, B., Quéma, V., and Roth, M. (2013). Traffic Management: A Holistic Approach to Memory Placement on NUMA Systems. In *Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, pages 381–393.

[Feliu et al. 2012] Feliu, J., Sahuquillo, J., Petit, S., and Duato, J. (2012). Understanding Cache Hierarchy Contention in CMPs to Improve Job Scheduling. In *International Parallel and Distributed Processing Symposium (IPDPS)*.