Impacto do Mapeamento de *threads* na Degradação do Processador*

Thiarles Soares Medeiros[1], **Arthur Francisco Lorenzon**[1]

[1]Laboratório de Otimização de Sistemas – Universidade Federal do Pampa
Av. Tiaraju, 810 – CEP 97.546-550 – Alegrete – RS – Brazil

{thiarlesmedeiros,aflorenzon}@unipampa.edu.br

1. Introdução

Com o crescente número de unidades de processamento a dissipação de calor se tornou um problema significativo, acelerando o processo de envelhecimento dos componentes físicos, reduzindo a vida útil e tornando-os mais suscetíveis a diferentes tipos de falhas. Com o aumento da temperatura dos núcleos (*core*) ativos os núcleos próximos são afetados, aumentando a variação da temperatura. Portanto, o envelhecimento do processador não está relacionado apenas a quantas threads estão executando, mas também a como elas são distribuídas pelos núcleos disponíveis.

Na literatura, dois tipos de abordagens já foram propostas. Uma delas explorando o impacto do TLP, ajustando artificialmente o número de threads durante a execução e aplicando estratégias de afinidade/mapeamento (alocação de threads nos núcleos disponíveis) baseadas no desempenho e/ou energia, como apresentado em [Lorenzon and Beck Filho 2019] e [Lorenzon et al. 2019]. Outra abordagem explora a afinidade/mapeamento de threads apenas considerando o envelhecimento. Nenhuma delas avaliou a influência do TLP e da afinidade/mapeamento das threads no envelhecimento. Considerando o cenário mencionado, avaliamos o impacto de diferentes graus de exploração de TLP e estratégias de alocação de threads (afinidade e mapeamento) no envelhecimento dos componentes de hardware.

2. Metodologia

Treze aplicações já paralelizadas e escritas em C/C++ a partir de diversos *benchmarks* foram escolhidas. Elas foram classificadas de acordo com a quantidade de acessos à memória e uso da CPU: Memory-bound e CPU-Bound. Elas estão implementadas em OpenMP, na qual fornece variáveis de ambiente para controlar a afinidade e mapeamento das threads. As políticas de mapeamento de threads definem em quais CPUs as threads devem ser alocadas: *threads*, *cores*, e *sockets*. Dentro de cada política de mapeamento, as threads podem ser distribuídas pelos núcleos, seguindo cinco estratégias de afinidade distintas: ***false***, ***true***, ***master***, ***close*** e ***spread***.

Os experimentos foram realizadas em duas arquiteturas multicore, AMD e Intel. As seguintes configurações foram avaliadas: (*i*) AMD de 16 cores - cinco graus de TLP e não foi utilizada a política de mapeamento *sockets*; e, (*ii*) Intel de 32 cores - seis graus de TLP e todas políticas de mapeamento. A frequência da CPU foi configurada para *ondemand* como *governor* do DVFS, que é o padrão usado na maioria das versões do

*Este trabalho foi parcialmente financiado pela FAPERGS nos projetos 19/2551-0001224-1 e 19/2551-0001689-1, CNPq PIBIC e FAPERGS PROBIC.

Linux. Compilamos as aplicações com o gcc/g++ 9.1, usando a opção de otimização -O3. Cada aplicação foi executada com o conjunto de entrada padrão.

No processador Intel, obtemos a temperatura da CPU a cada segundo diretamente do contador de hardware em tempo de execução. Quanto ao processador AMD a temperatura do núcleo é obtida por meio de simulação através dos valores de potência utilizada. Calculamos o processo de envelhecimento com base na temperatura operacional através da equação de Arrhenius, conforme definido em [Storino 2004].

3. Resultados

Por haver comunicação entre as threads nas aplicações MEM-Bound o número de threads e as estratégias de mapeamento e afinidade utilizadas durante a execução de uma aplicação desempenham um papel decisivo no envelhecimento do processador. Portanto, os melhores resultados foram alcançados na execução com um número de threads menor que o máximo disponível combinado às estratégias que (*i*) facilitam a comunicação entre as threads, alocando-as próximas uma das outras; (*ii*) ou as que distribuem as threads entre os núcleos para não sobrecarregar os recursos compartilhados. Os resultados para Intel e AMD apresentaram configurações distintas para os melhores casos. Os piores resultados foram alcançados quando a afinidade de threads foi definida como *master*, isso porque as threads são alocadas no mesmo local da thread principal. Com a configuração certa é possível reduzir o envelhecimento do processador em até 38% comparado à execução padrão do OpenMP. Também foi possível observar casos em que duas estratégias de afinidade de threads forneceram envelhecimento semelhante para todo o processador, mas envelheceram cada núcleo em razões distintas. Isto ocorreu devido ao fato que uma estratégia alocou as threads em núcleos específicos enquanto a outra não, reduzindo a possibilidade de criar pontos de calor.

Diferentemente das aplicações MEM-Bound, o envelhecimento do processador nas aplicações CPU-Bound depende mais do número de threads simultâneas em execução que da política de afinidade e mapeamento de threads. Nos processadores Intel e AMD, quanto maior o número de threads, menor o envelhecimento do processador. Embora a maioria das configurações executadas com 32 threads no processador Intel tenha apresentado resultados semelhantes (diferença de 4%), o menor envelhecimento do processador foi alcançado com o par *Sockets-close*. Ao comparar com a execução padrão de aplicações OpenMP, o envelhecimento do processador é 11% menor. O mesmo comportamento, mas a taxas distintas, pode ser observado no processador AMD de 16 núcleos.

References

Lorenzon, A. F. and Beck Filho, A. C. S. (2019). *Parallel Computing Hits the Power Wall*. SpringerBriefs in Computer Science. Springer International Publishing.

Lorenzon, A. F., de Oliveira, C. C., Souza, J. D., and Beck, A. C. S. (2019). Aurora: Seamless optimization of openmp applications. *IEEE Transactions on Parallel and Distributed Systems*, 30(5):1007–1021.

Storino, S. N. (2004). Method and apparatus for estimating remaining life of a product. US Patent 6,775,624.