Avaliação do Paralelismo dos Kernels EP e CG em Sistemas Embarcados

Lucas Marchesan Cunha, Renato B. Hoffmann, Dalvan Griebler, Isabel H. Manssour

[1] Escola Politécnica, Pontifícia Universidade Católica do Rio Grande do Sul (PUCRS)
Porto Alegre – RS – Brasil

(l.marchesan,renato.hoffmann)@edu.pucrs.br,(dalvan.griebler)@pucrs.br

Resumo. *Nesse artigo, testamos o ganho de desempenho obtido ao implementar códigos com processamento paralelo em sistemas embarcados genéricos. Para analisar ao desempenho em relação ao Speedup ideal, foram testados dois algoritmos (EP e CG) paralelos em dois sistemas embarcados diferentes. Os resultados mostram uma discrepância entre o melhor (3.98X) e o pior (1.38X) desempenho obtidos, indicando o tamanho do espectrum de desempenho.*

1. Introdução

Desde seu advento, a paralelização de programas tornou-se uma prática essencial para garantir o mais alto nível de desempenho possível de uma máquina moderna. Com uma implementação adequada, um código paralelo faz uso de todos os recursos disponíveis na arquitetura alvo, tendo um desempenho fortemente relacionado com a quantidade de núcleos do computador. Com isso, analisar seu desempenho com computadores mais robustos é de suma importância, para que seja possível discernir se o custo computacional de implementar a paralelização vale a pena em relação ao ganho de velocidade providenciado pela mesma (dado o seu numero menor de núcleos em comparação a sistemas maiores).

As máquinas que foram usadas para a realização dos experimentos são classificadas como sistemas embarcados. Elas foram escolhidas com base em sua portabilidade e estrutura simples em relação a aparelhos mais complexos. Um sistema embarcado é uma combinação de hardware e software cujo design é criado para que o aparelho realize uma função específica [Ganssle and Barr 2003]. Em outras palavras, esse tipo de sistema embarcado está anexado diretamente aos processadores. Os computadores utilizados na realização dos experimentos, Odroid-N2 e Tinker, são relativamente genéricos, sendo capazes de emular o ambiente de trabalho de um computador normal em uma escala menor e menos potente. Vale mencionar que a Odroid-N2 possui um maior poder de processamento em relação a Tinker. A grande vantagem é seu baixo consumo energético.

Para realizar a comparação do desempenho do processador é necessário usar códigos que realizem uma série de cálculos e funções, para que seus processamentos consumam um tempo que possa ser considerado como prático para ser paralelizado. Para isso, foram usados dois algoritmos, o primeiro deles é o 'EP', ou '*Embarassiling Parallel*', que faz diversos cálculos com os valores aleatórios gerados contidos em um vetor cujo tamanho é o parâmetro de entrada. Já o segundo deles é o 'CG', ou '*Conjugrate Gradient*', que, a partir de dois números (A,B) cria uma matriz de tamanho A X B e realiza uma série de operações com cada elemento da matriz seguindo as operações descritas no método do Gradiente Conjugado, que por sua vez tem como propósito resolver sistemas lineares de

equações. Ambos os códigos foram obtidos prontos do NPB-NAS, e a principal diferença entre eles é que o EP possui paralelismo sem dependência de dados enquanto que CG possui muitas sincronizações e uma pior localidade de acesso aos dados.

Essas aplicações foram executadas em dois sistemas embarcados para analisarmos o seu desempenho e, principalmente, verificar se é eficaz usar paralelismo em aplicações nesse tipo de sistema. Ou seja, o propósito desse trabalho é testar a eficácia de implementações paralelas em sistemas com pouco poder computacional e concluir se a melhora de desempenho adquirida compensa o custo de criar e operar as *threads*.

O restante do artigo foi organizado como se segue. Na seção 2, são apresentados artigos com temas e tópicos similares, assim como as diferenças e peculiaridades que diferencia esse projeto dos demais. Em seguida, na seção 3 são explicados os métodos e ambientes, entre outros detalhes mais específicos, utilizados para conduzir os experimentos realizados. Por fim, a seção 4 visa providenciar uma última análise e interpretação dos dados adquiridos durante o artigo, assim como algumas considerações finais.

2. Trabalhos Relacionados

No artigo [Belloch et al. 2022], os autores realizam uma pesquisa de desempenho que avalia a qualidade de áudio, a capacidade computacional e eficiência energética de vários equalizadores gráficos rodando em um celular que implementa um sistema 4x embarcado. Eles descrevem que os equalizadores gráficos corrigem ou melhoram o espectro de um sinal para atingir um certo objetivo. O principal diferencial deste trabalho é o objeto de pesquisa, pois é mais focado no desempenho dos equalizadores, enquanto a nossa pesquisa é focada no paralelismo resultante dos códigos executados.

Já o trabalho de [Jiang et al. 2023], tem como seu principal foco os *Spin Locks*, comumente usados para coordenar a ordem de execução de processos mutuamente exclusivos em um sistema embarcado. Nesse caso, a ideia é testar a eficiência desses *Spin Locks* em um programa paralelo, já que tal mecanismo possui uma série de experimentos usando programas sequenciais, mas a quantidade de experimentos paralelos ainda é relativamente escassa. Em relação à pesquisa deles, que analisa a eficiência de um mecanismo específico dentro da máquina, nosso trabalho visa estudar o desempenho do sistema como um todo.

Por fim, [Xu et al. 2022] trabalham com um sistema embarcado customizado que possui um relógio com *loop* restringido por fase, uma interface receptora e emissora universal e assíncrona, funções trigonométricas, entre outras utilidades. Esse hardware é utilizado para executar aplicações paralelas com uma escala de tempo em milissegundos que fazem uso de um NMPC (*nonlinear model-predictive control*). A principal diferença é que, apesar de ambos os trabalhos usarem sistemas embarcados, no trabalho deles, a aplicação paralela é secundária, enquanto esse aspecto é o foco dessa pesquisa.

3. Experimentos

Os experimentos foram realizados em dois sistemas embarcados distintos, o Odroid-N2 e o Tinker. O Odroid-N2 possui dois processadores, um 4x Cortex-A73 e um Dual-Core Cortex-A53, e tem como sistema operacional o Ubuntu 20.04. Já o Tinker possui dois processadores, o primeiro 4x ARM Cortex-A53 e o segundo 2x ARM Cortex-A72 e possui Linux linaro-alip 4.4.194 como seu sistema operacional.

Para realizar os experimentos necessários para medir o desempenho, os códigos foram paralelizados usando três interfaces de paralelismo distintas, sendo elas FastFlow, OpenMP e OneTBB (com as versões de FastFlow e OneTBB tendo sido obtidas em [Löff et al. 2021]), com o propósito de fornecer uma variedade maior de resultados. Assim, foi criado um *script* a partir do qual foram executadas cinco iterações de ambos os códigos CG (com dados de entrada fornecidos a partir da classe A do *NAS parallel benchmarks* [Bailey et al. 1991]) e EP (usando a classe B do *NAS parallel benchmarks*)rodando de uma à quatro threads, obtendo um total de 20 resultados para cada algoritmo tanto na Odroid-N2 quanto na Tinkerer. Vale mencionar também que o escalonamento das threads nesses testes foi feito pelo sistema operacional. Os gráficos das Figuras 1 e 2 ilustram o desempenho de cada um dos experimentos, onde o eixo x representa o número de *threads* utilizadas na execução e o eixo y representa o tempo de execução em segundos. Como podemos ver, quanto mais acentuada a curva, melhor o *speedup*.Esses gráficos também mostram os dois extremos, Odroid EP com uma curva mais suave e Tinker CG com um declínio pequeno de 3 para 4 *threads*.

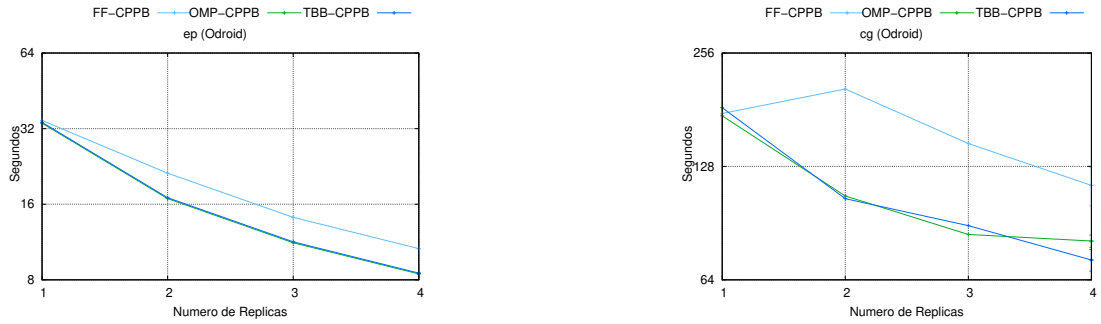

Figura 1. Tempos de execução da máquina Odroid-N2

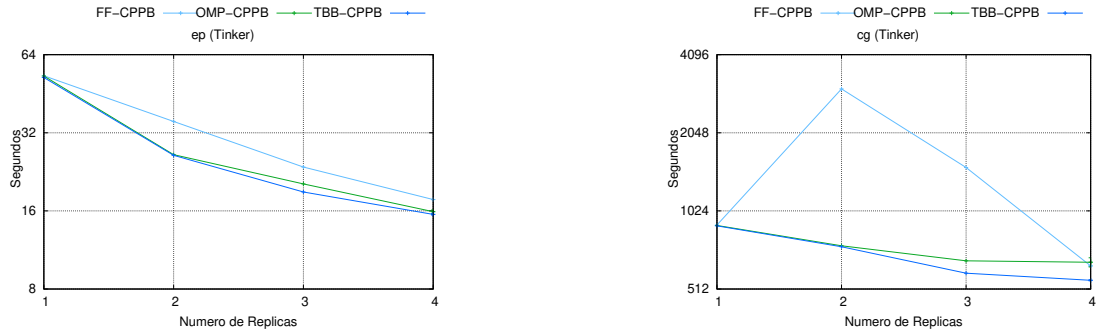

Figura 2. Tempos de execução da máquina Tinker

Na tabela 1, estão os dados extraídos da interface OpenMP como mostrados nos gráficos, com a adição do *Speedup* apresentado em cada um desses cenários. Nesse caso, o *Speedup* pode ser usado como uma métrica para medir o desempenho do processo de paralelização em cada código rodado em cada máquina. Lembrando que, como o número de *threads* testadas vai de um a quatro, quanto maior for o *Speedup*, melhor foi o desempenho da versão paralelizada.

A aplicação EP é a que apresenta o melhor desempenho em comparação à CG, assim como a máquina Odroid mostra ser mais compatível com o processo de paraleliza-

Nome do teste	Tempo com 1 Thread	Tempo com 4 Thread	*Speedup*
Tinker CG	899.702s (5.86759)	649.282s (26.5095)	1.38X
Tinker EP	52.936s (0.421312)	15.894s (0.238881)	3.33X
Odroid CG	174.344s (11.0126)	81.124s (3.03881)	2.15X
Odroid EP	33.642s (0.150785)	8.446s (0.032)	3.98X

Tabela 1. *Speedup* apresentado nos experimentos com OpenMP

ção até quatro *threads* em relação à Tinker. Com isso, podemos ver os dois extremos do *Speedup*, com o menor deles sendo o código CG executado no sistema Tinker com um *Speedup* de 1.38X e o maior sendo o código EP executado no sistema Odroid com um *Speedup* quase linear de 3.98X. Além disso, vale ressaltar que as interfaces OpenMP e OneTBB apresentaram desempenhos similares ao longo dos experimentos, com OpenMP sendo levemente melhor nas ultimas *threads* do CG. Em contraste, além de se mostrar mais lenta que as outras, FastFlow também apresentou um grande aumento de tempo rodando o código CG com 2 *threads*. Tais exemplos mostram o grande espectro de desempenhos que a implementação do paralelismo no código em um sistema embarcado possui, dependendo da quantidade de recursos da máquina e da natureza do código a ser executado.

4. Conclusões

Neste trabalho, foram explorados conceitos relacionados a sistemas embarcados e o que diferencia-os de outras máquinas, além de paralelismo, seus usos, suas vantagens e os tipos de aplicações sobre as quais deve ser implementado. Já nos resultados, podemos observar a grande diferença de desempenho entre o melhor caso, onde tanto a aplicação quanto o hardware favorecem a implementação paralela (com *Speedup* de 3.98X), e do caso oposto (com *Speedup* de apenas 1.38X). Para dar sequência a esse projeto, pode-se expandir tanto o número de sistemas embarcados diferentes quanto o número de aplicações testadas, que seria ideal para aumentar o espectrum de *Speedups* e comparações, já que os resultados obtidos apresentaram os dois extremos de desempenho, sendo esses o *Speedup* de 1.38X e de 3.98X.

Referências

Bailey, D. H., Barszcz, E., Barton, J. T., Browning, D. S., Carter, R. L., Dagum, L., Fatoohi, R. A., Frederickson, P. O., Lasinski, T. A., Schreiber, R. S., Simon, H. D., Venkatakrishnan, V., and Weeratunga, S. K. (1991). The nas parallel benchmarks summary and preliminary results. In *Supercomputing '91:Proceedings of the 1991 ACM/IEEE Conference on Supercomputing*, pages 158–165.

Belloch, J. A., Badía, J., León, G., Bank, B., and Välimäki, V. (2022). Multicore implementation of a multichannel parallel graphic equalizer. *J. Supercomput.*, 78(14):15715–15729.

Ganssle, J. and Barr, M. (2003). *"Embedded Systems Dictionary*. CRC Press.

Jiang, X., Chen, Z., Yang, M., Guan, N., Tang, Y., and Wang, Y. (2023). A unified blocking analysis for parallel tasks with spin locks under global fixed priority scheduling. *IEEE Transactions on Computers*, 72(1):15–28.

Löff, J., Griebler, D., Mencagli, G., and et al. (2021). The nas parallel benchmarks for evaluating c++ parallel programming frameworks on shared-memory architectures. *Future Generation Computer Systems*, 125:743–757.

Xu, F., Guo, Z., Chen, H., Ji, D., and Qu, T. (2022). A custom parallel hardware architecture of nonlinear model-predictive control on fpga. *IEEE Transactions on Industrial Electronics*, 69(11):11569–11579.