Análise da Escalabilidade de Aplicações Paralelas em Sistemas Embarcados Multiprocessados na Borda*

Ueslei B. Brandt[1], Marcelo C. Luizelli[1], Fábio D. Rossi[2], Arthur F. Lorenzon[1]

[1]Universidade Federal do Pampa – Alegrete – RS – Brazil

[2]Instituto Federal Farroupilha - Campus Alegrete – Alegrete – RS – Brasil

uesleibrandt.aluno@unipampa.edu.br

Resumo. *Com o constante aumento da utilização de sistemas embarcados na borda, a paralelização de aplicações desses sistemas vem com a intenção de melhorar seu desempenho. Nesse sentido, cresce a importância de analizar a escalabilidade dessas aplicações. Assim este trabalho objetiva analizar a escalabilidade de aplicações paralelas executadas na borda. Através da execução de aplicações com diferentes características, mostramos que, enquanto aplicações CPU-intensivas escalam perfeitamente de acordo com o número de threads, aplicações com comunicação e sincronização entre as threads não escalam bem no co-processador Epiphany da Parallella Board.*

1. Introdução

Sistemas embarcados utilizados na borda (*edge*) têm sido construídos sobre processadores multicore para eficientemente atender às demandas de aplicações por mais poder computacional. Além disso, existe uma preocupação em reduzir o consumo de energia, uma vez que a maioria dos dispositivos encontrados na borda possuem restrições energéticas. Portanto, dado o aumento do número de aplicações sendo migradas para a borda, otimizar o uso dos recursos computacionais de tais sistemas é de extrema importância.

O aumento no desempenho das aplicações pode ser atingido com a exploração do paralelismo no nível de *threads* (*Thread-Level Parallelism* – TLP) [Lorenzon and Beck Filho 2019]. Neste caso, várias unidades de processamento (e.g., núcleos) executam simultaneamente diferentes partes do mesmo programa com o objetivo de reduzir o tempo total de execução. Diferentes trabalhos têm avaliado a exploração de TLP na borda. [Raase and Nordström 2015] utilizam o método *Lattice Boltzmann* para avaliar o desempenho da fluidodinâmica computacional em computação de borda. [Guillén et al. 2021] avaliam o desempenho e consumo de energia da computação de borda comparado com computação em nuvem. [Medeiros et al. 2020] avalia o comportamento de aplicações paralelas executando na borda com relação ao envelhecimento do processador. [Femminella et al. 2016] analizam os diferentes comportamentos de uma implementação do *Hadoop* em um ambiente físico usando computadores de propósito geral com dois tipos diferentes de virtualizações hospedadas em uma nuvem Openstack. Adicionalmente, [Lorenzon et al. 2015] avalia o desempenho e consumo de energia de aplicações paralelas em sistemas embarcados.

Considerando o exposto acima, este trabalho apresenta uma análise da escalabilidade de aplicações paralelas que executam em sistemas embarcados da borda. Para tanto, quatro aplicações paralelas com diferentes características relacionadas à exploração do

*Este trabalho foi parcialmente financiado pela FAPERGS nos projetos 19/2551-0001224-1, 19/2551-0001689-1 e 17/2551-0001193-7 e PROBIC-FAPERGS

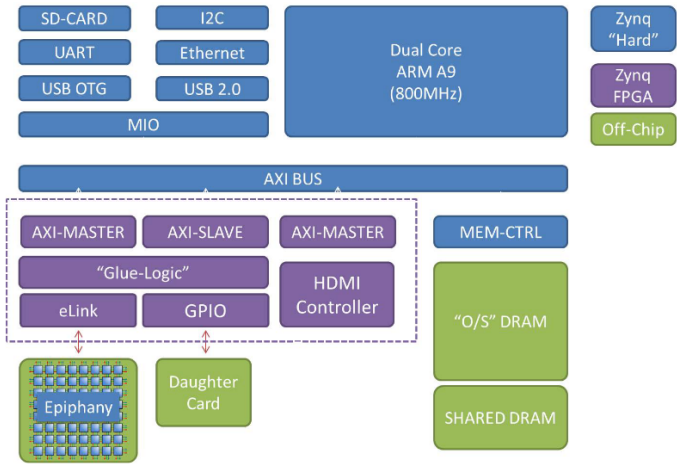

Figura 1. Visão de alto nível da arquitetura da Parallella Board

paralelismo foram executadas com diferentes números de *threads* (de 1 até 16) no co-processador embarcado *Epiphany*. Nossos resultados mostram que para aplicações CPU-intensivas, a escalabilidade é próxima do ideal, com speedup de 15.8 $\times$ na execução com 16 *threads*, em relação à aplicação sequencial. Por outro lado, para aplicações onde há comunicação e sincronização entre as threads, a escalabilidade é limitada. Por exemplo, a aplicação *Game of Life* teve um speed up de apenas 2.9 $\times$ na execução com 16 *threads*, em relação à versão sequencial.

2. Sistemas Embarcados Multiprocessados na Borda

Dispositivos da Internet das Coisas (IoT - *Internet of Things*), que são sistemas inteligentes com conexão à nuvem, estão em franca expansão e têm aplicações em diversas áreas. Entretanto, estes sistemas necessitam cada vez mais de respostas rápidas e com baixa latência e, por este motivo, o processamento computacional vem sendo progressivamente transferido da nuvem (*cloud*) para a borda. Os computadores utilizados na borda são normalmente caracterizados por serem embarcados e com restrições de desempenho e consumo de energia, sendo a empresa ARM uma das principais desenvolvedoras de chips para tais sistemas.

Neste trabalho, utilizamos a Parallela board com seu processador princial, Dual-core ARM® Cortex™-A9 com a arquitetura ARMv7-A e 800MHz, com foco principal no co-processador *Epiphany* com a arquitetura 32-bit *dual-issue superscalar RISC* de 16 núcleos cada núcleo com 1GHz. Ela é um computador utilizado pra alto desempenho, principalmente para o uso em computação paralela de borda [1]. A Figura 1 apresenta uma visão de alto nível da arquitetura da *Parallella Board*. Na parte superior da Figura 1 podemos ver o processador ARM A9 com 2 núcleos. O co-processador de 16 núcleos *Epiphany* aparece no canto inferior esquerdo da imagem.

Quando uma aplicação é executada neste sistema, ela começa sua execução no processador principal (*dual-core* ARM Cortex-A9), onde são carregados todos os dados da aplicação. Quando a aplicação encontra um ponto de *offloading* (definido pela diretiva de offload do OpenMP), os dados da região paralela são enviados para o co-processador Epiphany. Uma vez no Epiphany, os dados são computados em paralelo de acordo com o número de *threads* definido pela variável de ambiente do OpenMP (OMP_NUM_THREADS).

[1] disponível em https://parallella.org/

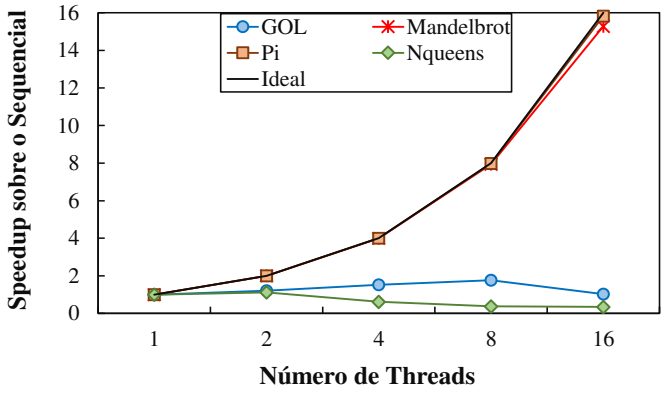

Figura 2. Aceleração (*Speedup*) de cada aplicação (*eixo y*) com diferentes números de *threads* (*eixo x*). Quanto maior, melhor.

3. Metodologia

Nós utilizamos quatro aplicações já paralelizadas com OpenMP e com as funções de *offload* para a arquitetura Parallella[2]: *Game of Life*, *Mandelbrot Set*, *N-queens* e *Cálculo do Pi*. Cada aplicação foi executada com a entrada padrão configurada no conjunto de aplicações. Os experimentos foram realizados na *Parallella Board* utilizando seu co-processador *epyphany* (descrito na Seção 2). Cada aplicação foi compilada com o compilador *ompicc* na versão 2.0.0., que é um compilador da Linguagem C/C++ para a arquitetura em questão com bibliotecas de OpenMP ja inseridas em sua implementação. O Sistema Operacional utilizado foi o Linux Ubuntu na versão 15.04 com kernel v. 4.6.0+. Nós executamos as aplicações com diferentes números de *threads*: 1, 2, 4, 8 e 16. Cada configuração (aplicação e número de *threads*) foi executada 10 vezes e os resultados apresentados na próxima seção compreendem a média das execuções. Por fim, para configurar o numero de *threads* utilizado pela aplicação, nós utilizamos a variável de ambiente OMP_NUM_THREADS.

4. Resultados Experimentais

Nesta seção, discutimos os resultados relacionados a desempenho das aplicações considerando as seguintes métricas: *speedup*, que mede a aceleração da versão paralela com relação a versão sequencial; e *eficiência*, que mostra o quão eficiente a aplicação é com relação ao uso dos recursos computacionais (e.g., número de núcleos). Assim, a Figura 2 apresenta o *speedup* (*eixo y*) para cada aplicação quando executada com diferentes números de *threads* (*eixo x*). Por exemplo a execução da aplicação Pi com 16 núcleos teve um speedup de 15,82 vezes.

Para as aplicações de calculo do Pi e *Mandelbrot Set* pode-se observar uma constante melhora no *speedup* conforme o número de *threads* aumenta. Isso acontece pois, essas aplicações são CPU-intensivas, fazendo elas não sofrerem com a comunição entre as *threads*. Por outro lado, para as aplicações *N-queens* e GOL pode-se observar que aumentar o número de *threads* nem sempre leva a uma melhora de desempenho: para *N-queens*, o melhor desempenho é atingido com 2 threads, enquanto que para o GOL, 8 threads apresenta o melhor resultado. Este comportamento se dá pois estas aplicações possuem comunicação e sincronização entre as *threads*, se tornando um gargalo no desempenho.

A Figura 3 apresenta a eficiência (*eixo y*) para cada número de *threads* (*eixo x*). Quanto mais próximo de 1, mais eficiente é a execução da aplicação paralela, significando

[2]Os códigos utilizados estão disponíveis em: https://github.com/parallella

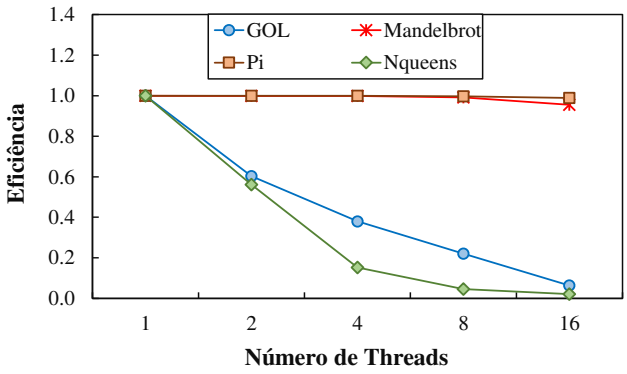

Figura 3. Eficiência das aplicações (eixo y) e número de threads (eixo x)

que o fator de melhora de desempenho é igual ou similar ao número de *threads* executando a aplicação. A Figura demonstra um comportamento similar ao observado na Figura 2, onde aplicações CPU-intensívas (Pi e Mandelbrot) tiveram uma eficiência proxima do ideal. Por outro lado, as aplicações com comunicação e sincronização entre as *threads* (N-queens e GOL) têm uma redução na eficiência conforme o número de *threads* aumenta.

5. Conclusão

Este trabalho apresentou uma análise de desempenho de aplicações paralelas em uma arquitetura de borda. Para tanto, executamos quatro aplicações no co-processador *Epiphany* da *Parallella board* com diferentes números de *threads*. Através dos experimentos, mostramos que aplicações CPU-instensivas tem escalabilidade próxima do ideal, ao contrário de aplicações que possuem comunição e sincronização entre as *threads* . Por fim, como trabalhos futuros, pretendemos explorar diferentes aplicações e também investigar o impacto do mapeamento de *threads* no respectivo co-processador.

Referências

Femminella, M., Pergolesi, M., and Reali, G. (2016). Performance evaluation of edge cloud computing system for big data applications. pages 170–175.

Guillén, M. A., Llanes, Antonio, I. B., Martínez-España, R., Bueno-Crespo, A., Cano, J.-C., and Cecilia, J. M. (2021). Performance evaluation of edge-computing platforms for the prediction of low temperatures in agriculture using deep learning. *The Journal of Supercomputing*, 77:818–840.

Lorenzon, A. F. and Beck Filho, A. C. S. (2019). *Parallel computing hits the power wall: principles, challenges, and a survey of solutions*. Springer Nature.

Lorenzon, A. F., Cera, M. C., and Schneider Beck, A. C. (2015). Performance and energy evaluation of different multi-threading interfaces in embedded and general purpose systems. *Journal of Signal Processing Systems*, 80(3):295–307.

Medeiros, T. S., Berned, G. P., Navarro, A., Rossi, F. D., Luizelli, M. C., Brandalero, M., Hübner, M., Beck, A. C. S., and Lorenzon, A. F. (2020). Aging-aware parallel execution. *IEEE Embedded Systems Letters*, 13(3):122–125.

Raase, S. and Nordström, T. (2015). On the use of a many-core processor for computational fluid dynamics simulations. *Procedia Computer Science*, 51:1403–1412. International Conference On Computational Science, ICCS 2015.