# Avaliando Paralelismo em Dispositivos com Recursos Limitados

**Renato B. Hoffmann, Dalvan Griebler**

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

renato.hoffmann@edu.pucrs.br,dalvan.griebler@pucrs.br

***Resumo.*** *Sistemas computacionais com recursos limitados dispõem cada vez mais recursos computacionais. Portanto, para atender às restrições de desempenho e obter um baixo consumo de energia, é necessário utilizar o paralelismo. Este trabalho avaliou 4 aplicações em 3 dispositivos diferentes comparando 5 interfaces de paralelismo.*

Um dos principais desafios em sistemas embarcados é atender às restrições de desempenho, geralmente tempo-real, da aplicação, ao mesmo tempo que mantém um baixo consumo de energia [Barr and Massa 2006]. Muitos sistemas embarcados possuem processadores com múltiplos núcleos e aceleradores como unidades gráficas de propósito geral. Portanto, o desenvolvedor de sistemas embarcados deve ser proficiente em programação paralela, o que implica na necessidade de escolher dentre a série de opções disponíveis. Sendo assim, a contribuição deste trabalho consiste na avaliação de alternativas para o processamento paralelo em dispositivos com recursos limitados.

Neste documento, definimos como recurso limitado qualquer dispositivo de computação equipado com um processador ARM Cortex. Existem duas razões: a tecnologia RISC ARM é conhecida por sua eficiência energética; a família Cortex fornece variantes para sistemas embarcados [P B et al. 2022]. Nesse caso, realizamos os experimentos com 3 dispositivos de baixo poder computacional: Nvidia Jetson Nano, Raspberry Pi 4 B e Odroid-N2. Outras versões de software são GCC 9.4.0, Intel TBB (2020.1) e FastFlow (3.0.0).

**Tabela 1. Vazão de todas aplicações de processamento de *stream*.**

| Dispositivo | Aplicação | Vazão (items/segundo) | | | | | |
|---|---|---|---|---|---|---|---|
| | | **Seq** | **SPar** | **Fastflow** | **TBB** | **Threads** | **OpenMP** |
| **Jetson** | Bzip2 | 4.21 | 13.09 | 13.07 | 13.94 | 13.80 | 13.80 |
| | Ferret | 18.57 | 64.58 | 63.20 | 73.39 | 73.54 | 73.11 |
| | Lane | 1.07 | 4.02 | 4.02 | 4.09 | 4.14 | 4.10 |
| | Face | 0.29 | 0.99 | 0.98 | 1.03 | 1.03 | 1.02 |
| **Raspberry** | Bzip2 | 4.70 | 8.23 | 8.26 | 8.97 | 8.91 | 8.96 |
| | Ferret | 25.34 | 53.53 | 51.91 | 65.94 | 64.40 | 63.74 |
| | Lane | 3.03 | 7.64 | 7.67 | 7.95 | 8.03 | 7.76 |
| | Face | 0.48 | 0.95 | 0.97 | 1.04 | 1.03 | 1.03 |
| **Odroid** | Bzip2 | 6.39 | 17.26 | 17.13 | 17.51 | 17.42 | 17.42 |
| | Ferret | 34.15 | 134.11 | 134.78 | 155.21 | 151.30 | 151.64 |
| | Lane | 4.00 | 15.94 | 15.42 | 16.13 | 15.95 | 15.86 |
| | Face | 0.48 | 1.80 | 1.82 | 1.90 | 1.89 | 1.90 |
| **Média Total** | | 0.3913 | 1.2403 | 1.2407 | 1.3194 | 1.3073 | 1.3079 |

Os experimentos foram realizados com quatro aplicações C++ de processamento de fluxo de dados. `Bzip2` para compressão de dados, `Ferret` para busca de similaridade entre imagens, `Lane` para detecção de pistas e `Face` para reconhecimento facial [Garcia et al. 2022]. Todas essas aplicações foram implementadas com as seguintes versões paralelas: SPar, FastFlow, TBB, Threads e OpenMP. É importante destacar que essas interfaces possuem diferentes níveis de programabilidade e abstração. Por exemplo, SPar e OpenMP utilizam anotações de código mais alto-nível, FastFlow e TBB utilizam esqueletos de algoritmo enquanto que Threads utiliza construções mais baixo nível do padrão da linguagem C++.

A Tabela 1 demonstra a vazão em itens por segundo, onde o maior desvio padrão observado foi $0,92$. Utilizando todos os recursos paralelos do dispositivo (4 núcleos), todas as versões paralelas tiveram um *speed-up* superior à 3. Considerando todas as aplicações, TBB obteve a melhor média total, enquanto que SPar foi pior. Já a Tabela 2 demonstra o consumo energético, onde o maior desvio padrão observado foi $0,00089$. Nesse caso, o resultado foi antagônico ao de vazão, uma vez que as interfaces que tiveram o melhor desempenho obtiveram um consumo de energia inferior. Isso é esperado, uma vez que uma vazão maior implica em uma melhor utilização dos recursos computacionais do dispositivo. Entretanto, as versões Threads e OpenMP desempenharam de forma similar. De forma geral, TBB se mostrou a melhor interface de paralelismo para essas aplicações. No futuro, buscaremos adicionar aplicações de paralelismo de dados assim como aplicações paralelas com GPGPU.

**Tabela 2. Consumo de energia todas as aplicações de processamento de *stream*.**

| Dispositivo | Aplicação | Consumo Total de Energia (mWh) | | | | | |
|---|---|---|---|---|---|---|---|
| | | **Seq** | **SPar** | **Fastflow** | **TBB** | **Threads** | **OpenMP** |
| **Jetson** | Bzip2 | 79.8 | 45 | 45.4 | 42.6 | 43 | 42.8 |
| | Ferret | 160.4 | 84.6 | 87.2 | 75.2 | 76.2 | 76 |
| | Lane | 64.8 | 28.4 | 28.2 | 27.8 | 27.6 | 27.2 |
| | Face | 93.4 | 50.2 | 50.2 | 48.8 | 48 | 48.2 |
| **Raspberry** | Bzip2 | 99.6 | 70.6 | 71.8 | 66.4 | 66 | 66.6 |
| | Ferret | 166.6 | 97.4 | 100.8 | 79.2 | 80.6 | 81.8 |
| | Lane | 37.8 | 18 | 18.2 | 17.4 | 17.2 | 17.6 |
| | Face | 82.4 | 50 | 49.2 | 46.4 | 45.8 | 46 |
| **Odroid** | Bzip2 | 54.8 | 35.8 | 36.2 | 35.8 | 34.6 | 35.6 |
| | Ferret | 88 | 47.4 | 47 | 41.2 | 42.4 | 41.8 |
| | Lane | 20.6 | 10.6 | 11.6 | 10.4 | 11 | 10.8 |
| | Face | 56 | 29.6 | 29.6 | 28.8 | 29 | 28.6 |
| **Média Total** | | 83.683 | 47.300 | 47.950 | 43.333 | 43.450 | 43.583 |

## Referências


Barr, M. and Massa, A. (2006). *Programming Embedded Systems: With C and GNU Development Tools*. O'Reilly Media, Inc., Boston, United States.

Garcia, A. M., Griebler, D., Schepke, C., and Fernandes, L. G. (2022). SPBench: a framework for creating benchmarks of stream processing applications. *Computing*, In press(In press):1–23.

P B, H., Anireddy, S. R., F T, J., and R, V. (2022). Introduction to arm processors & its types and overview to cortex m series with deep explanation of each of the processors in this family. In *2022 International Conference on Computer Communication and Informatics (ICCCI)*, pages 1–8.