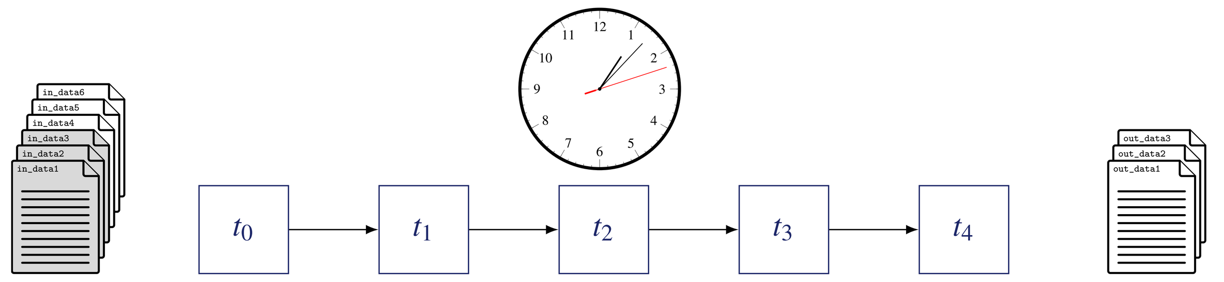
Efficient Task Scheduling for Streaming Apps on Heterogeneous SoCs

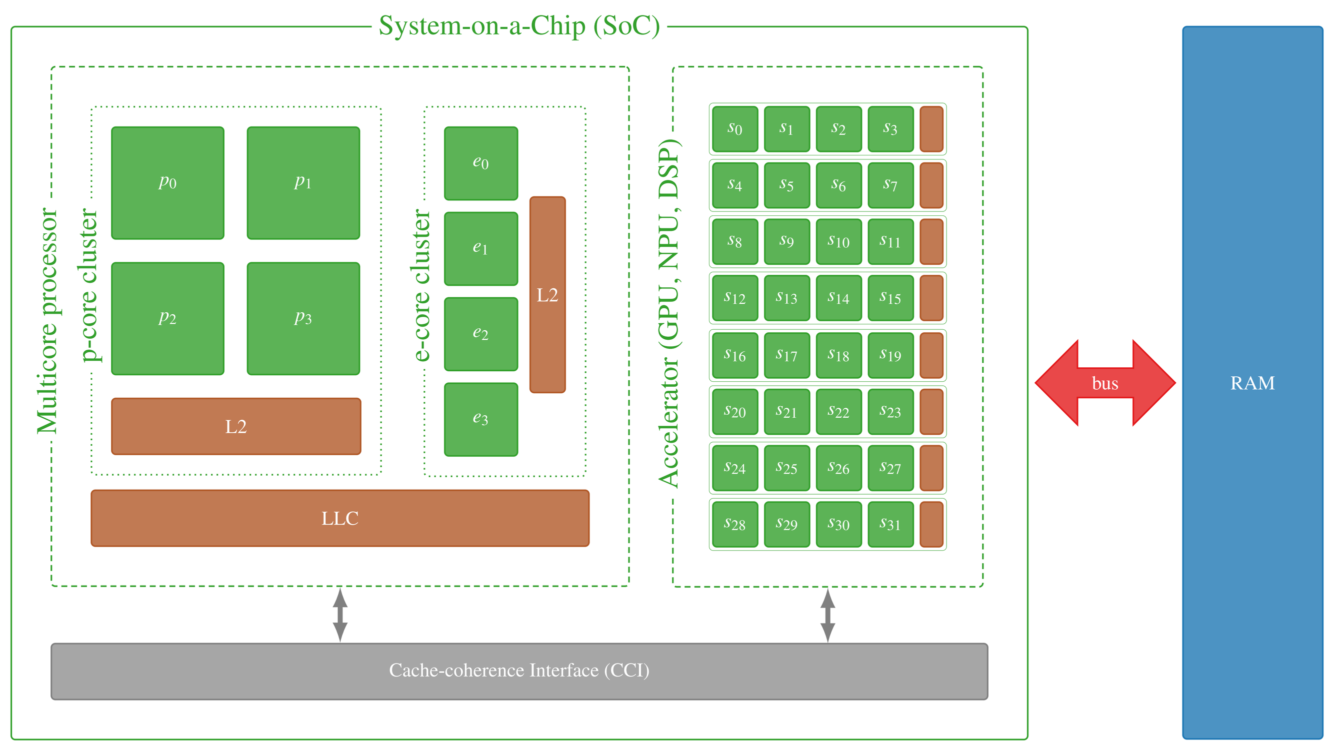
Yacine Idouar, Adrien Cassagne & Julien Sopena

— LIP6 Project 2023-2024 — Sorbonne Université & CNRS

Context: Streaming Apps & SoCs

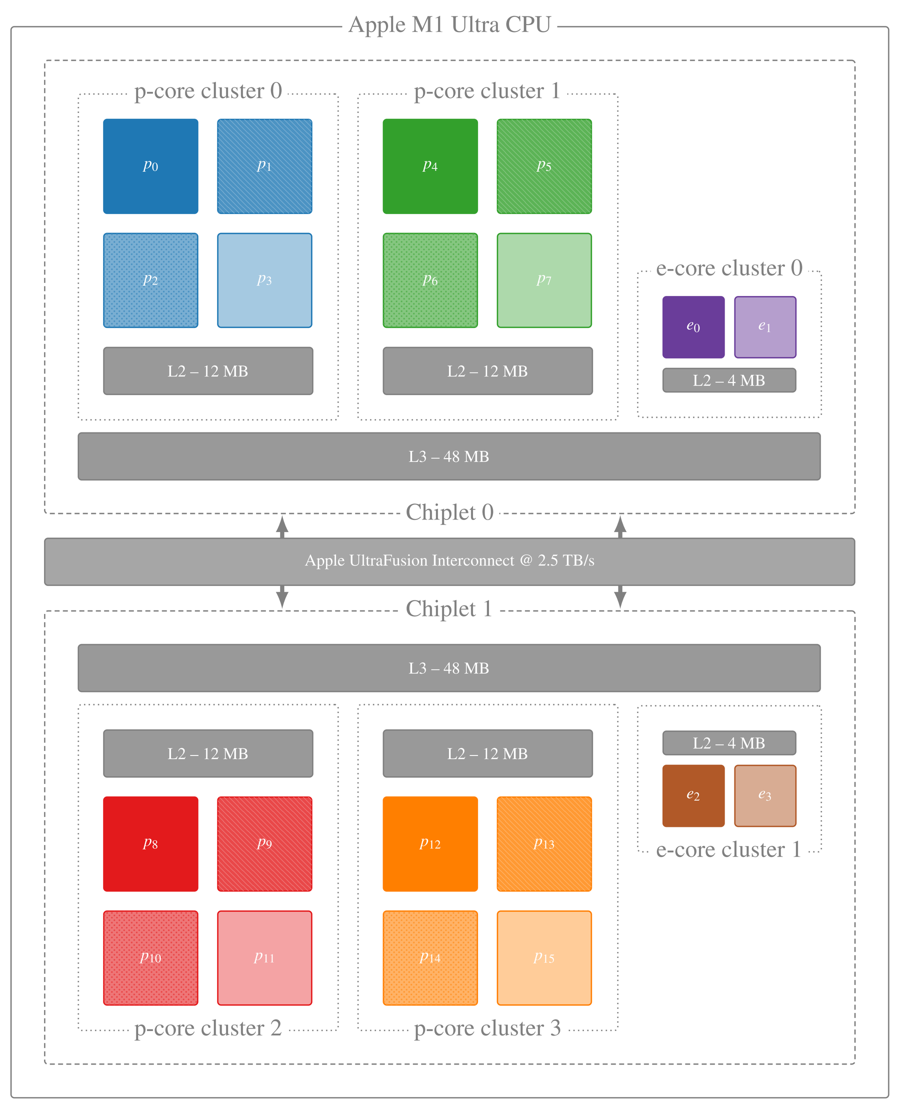


- **Large streams** of independent data & **efficiency constraints**
- **Stable computation pattern** (almost the same for each stream)
- Well-spread: **Digital communications**, video processing, DNN, ...



- **Heterogeneous** systems: **Powerful** and power **efficient** CPU cores
- **Specialized process units**: GPU, NPU, DSP, ...
- **Unified global memory**: Great opportunities for programming!

Targeted System: Apple M1 Ultra

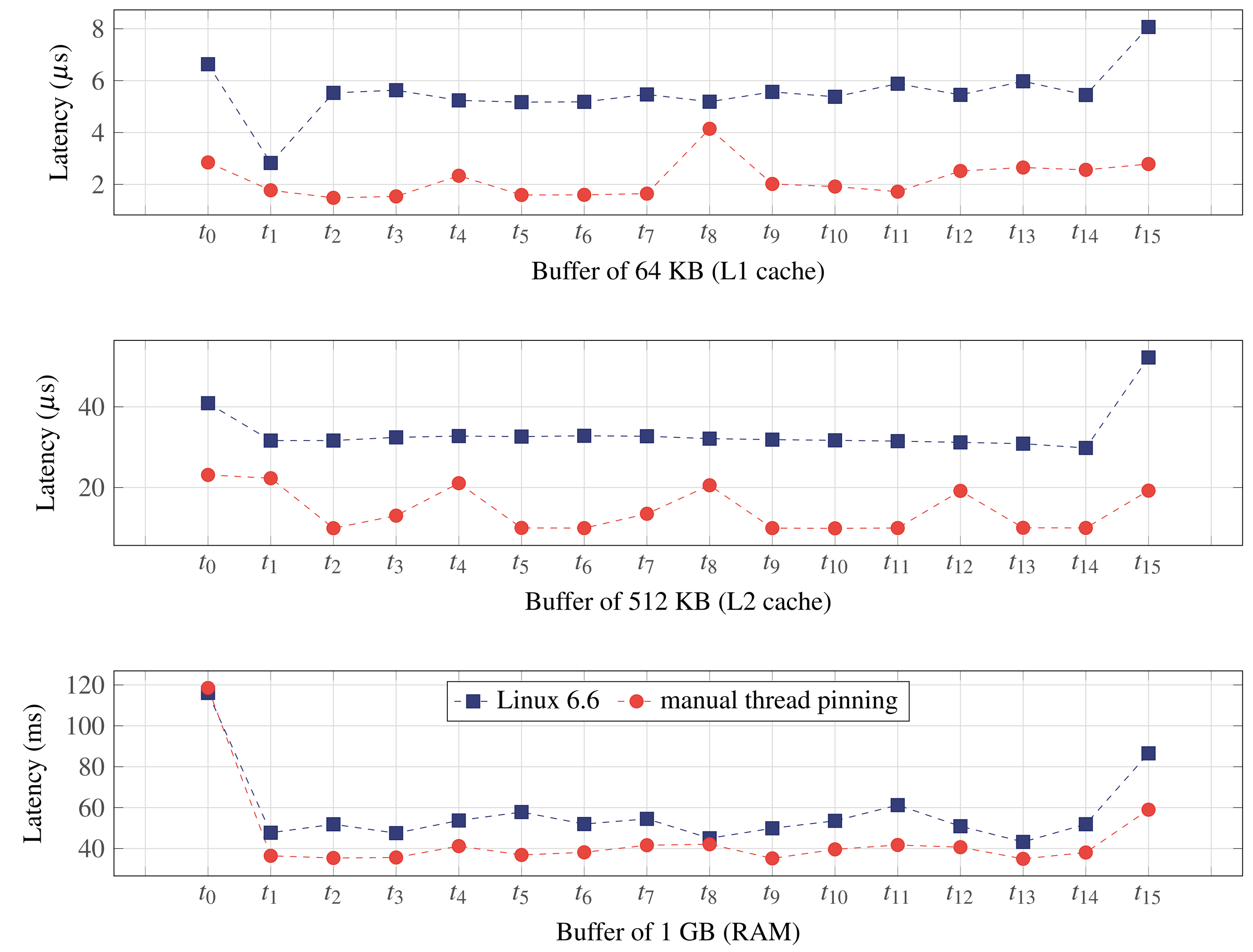


- **20 CPU cores**
  - 4 clusters of 4 p-cores @ 3.230 GHz
  - 2 clusters of 2 e-cores @ 2.064 GHz
  - ARMv8.5-A ISA with 128-bit NEON SIMD
- 5 nm TSMC chiplet design
  - Fusion of two M1 Max
- RAM bandwidth: 800 GB/s
- Fedora Asahi Linux OS
  - Kernel 6.6
  - **Thread pinning***

*: Running Linux is required as macOS does not provide a working thread pinning mechanism

Memory Bound Micro-benchmark

A chain of sixteen $t_i^{i \in [0..15]}$ tasks is considered. Each task performs streaming increments: $\mathcal{B} \leftarrow \mathcal{B} + 1$ where $\mathcal{B}$ is a buffer of size N. Each task is run on a single thread and mapped onto a **pipeline stage**. The communication between two consecutive stages is achieved through a $1 \rightarrow 1$ **producer-consumer algorithm** (from the StreamPU runtime [2]).

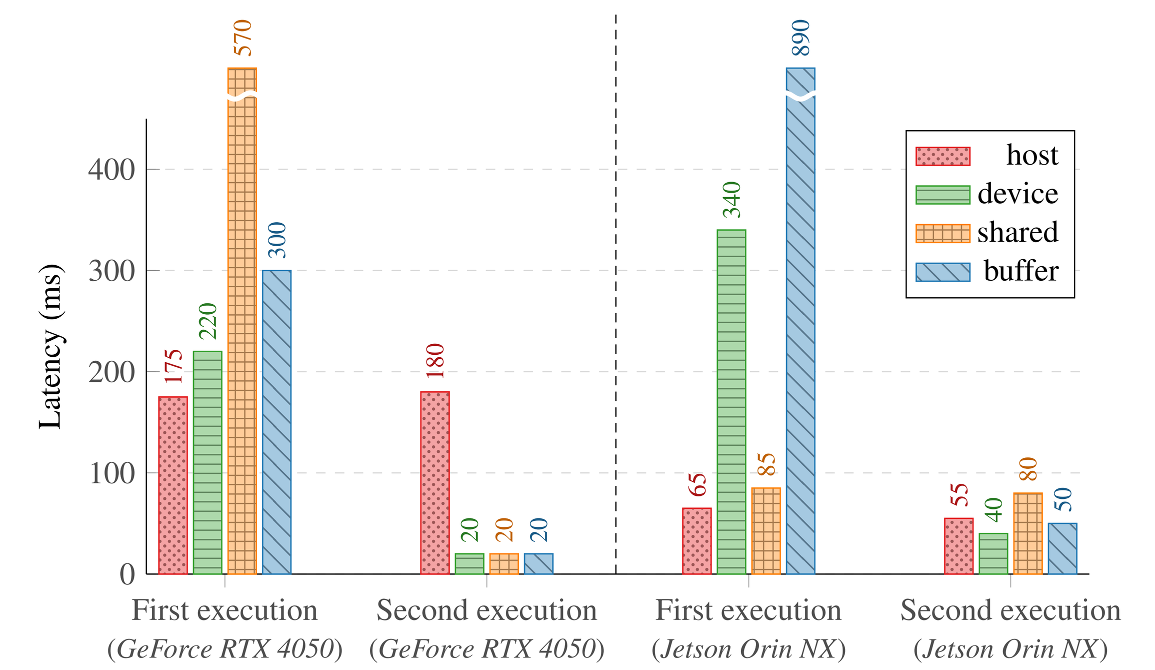


Per task latency depending on thread scheduling policy and buffer size N.

- **Linux scheduler is always outperformed** by manual thread pinning
  - Balanced workload on all the cores & useless thread migrations
- **Manual thread pinning** according to cores **physical locality**
  - Tasks are mapped to p-cores only: $p_i \leftarrow t_i$ (e-cores are left idle)

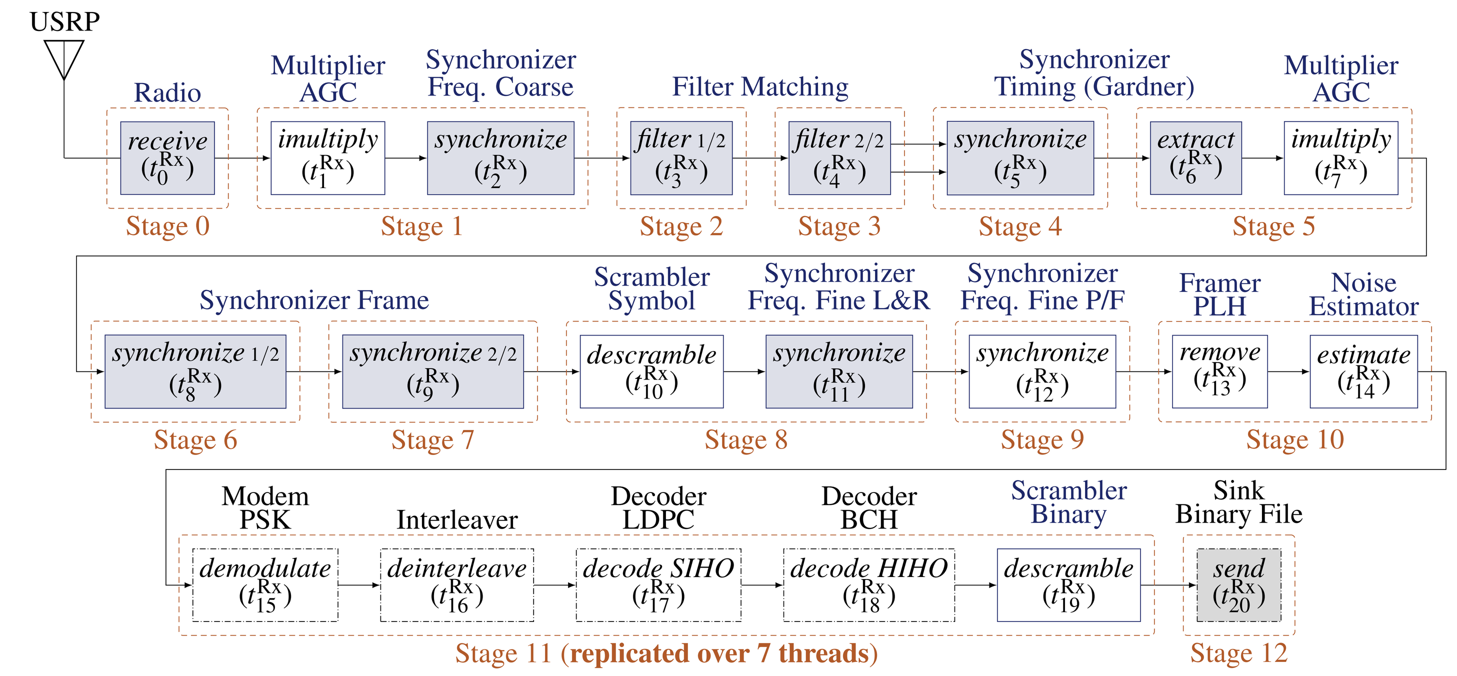
GPU Memory Allocation & Transfer Policies

Scenario of a **first exec of a simple kernel on GPU** (that may or may not **require a memory copy** depending on the selected memory policy) followed by a **second exec** of the same kernel (**no memory copy**) [3].



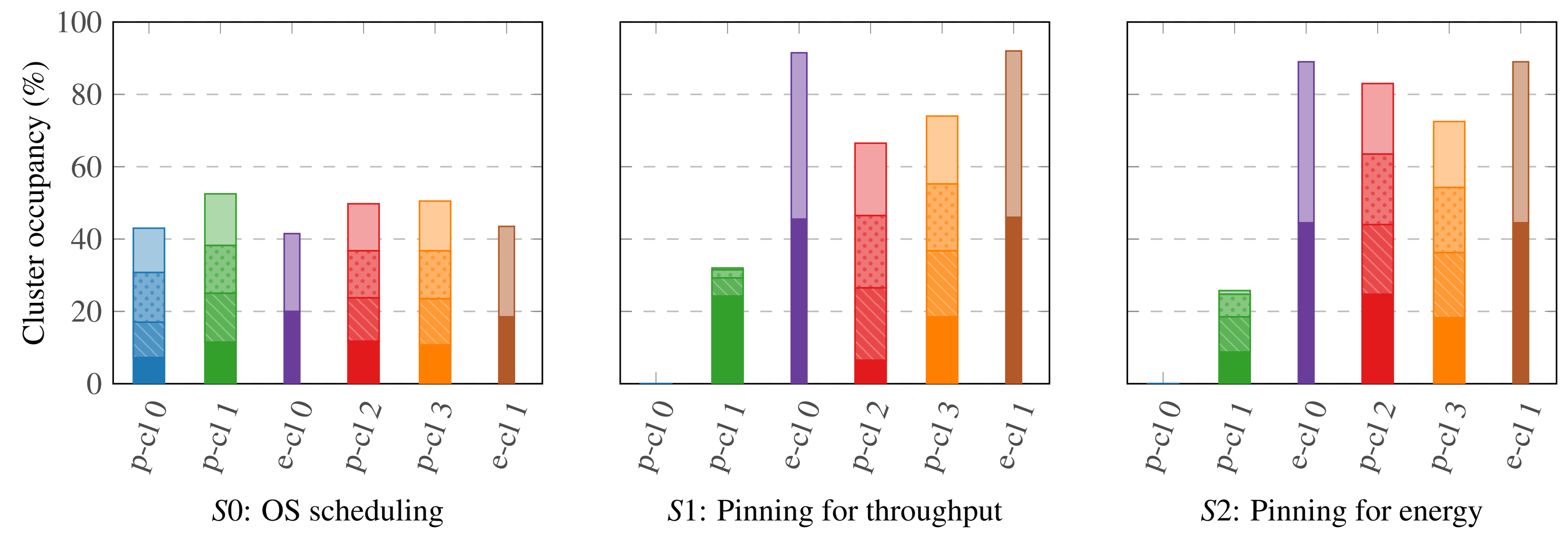
Effect of **SYCL memory policies** [4] on traditional discrete GPU architecture (*GeForce RTX 4050*) and on integrated GPU with unified memory (Jetson Orin NX).

Results on a Real-world Application



Task graph of the DVB-S2 Rx. Plain tasks cannot be replicated.

- **Digital Video Broadcasting – Satellite – 2nd Gen.** (DVB-S2)
  - Focus on the most compute intensive part: The **receiver** (Rx)
  - **Efficient SIMD implem**, 13-stage pipeline with **replication** [1]



Occupancy of the M1 Ultra CPU clusters depending on three different thread mapping strategies. *S0*: Linux 6.6 scheduler. *S1*: Manual thread pinning to maximize the app throughput. *S2*: Manual thread pinning to minimize the app energy consumption.

Strategy	Throughput (Mb/s)	Power (W)	Energy/fra (mJ)
S0	54.5	32	8.0
S1	56.0	30	7.3
S2	53.6	26	6.6

Compared to *S0* strategy
- *S1*: Throughput gain: +3% & Energy efficiency: +10%
- *S2*: Throughput gain: −1.5% & Energy efficiency: **+20%**

References

[1] A. Cassagne, M. Léonardon, R. Tajan, C. Leroux, C. Jégo, O. Aumage, and D. Barthou. A flexible and portable real-time DVB-S2 transceiver using multicore and SIMD CPUs. In International Symposium on Topics in Coding (ISTC). IEEE, Sept. 2021.

[2] A. Cassagne, R. Tajan, O. Aumage, D. Barthou, C. Leroux, and C. Jégo. A DSEL for high throughput and low latency software-defined radio on multicore CPUs. Wiley Concurrency and Computation: Practice and Experience, 35(23):e7820, July 2023.

[3] S. Joube, H. Grasland, D. Chamont, and E. Brunet. Comparing SYCL data transfer strategies for tracking use cases. Journal of Physics: Conference Series, 2438(1):012018, Feb. 2023.

[4] R. Reyes, G. Brown, R. Burns, and M. Wong. SYCL 2020: More than meets the eye. In International Workshop on OpenCL. ACM, 2020.

SCAN ME