ALMOS Operating System on TSAR Single-Chip cc-NUMA Many-Core

Ghassan Almaless

03 decembre 2012

Traditional Sources of Performance Are Dead, Long Live for Many-Cores!

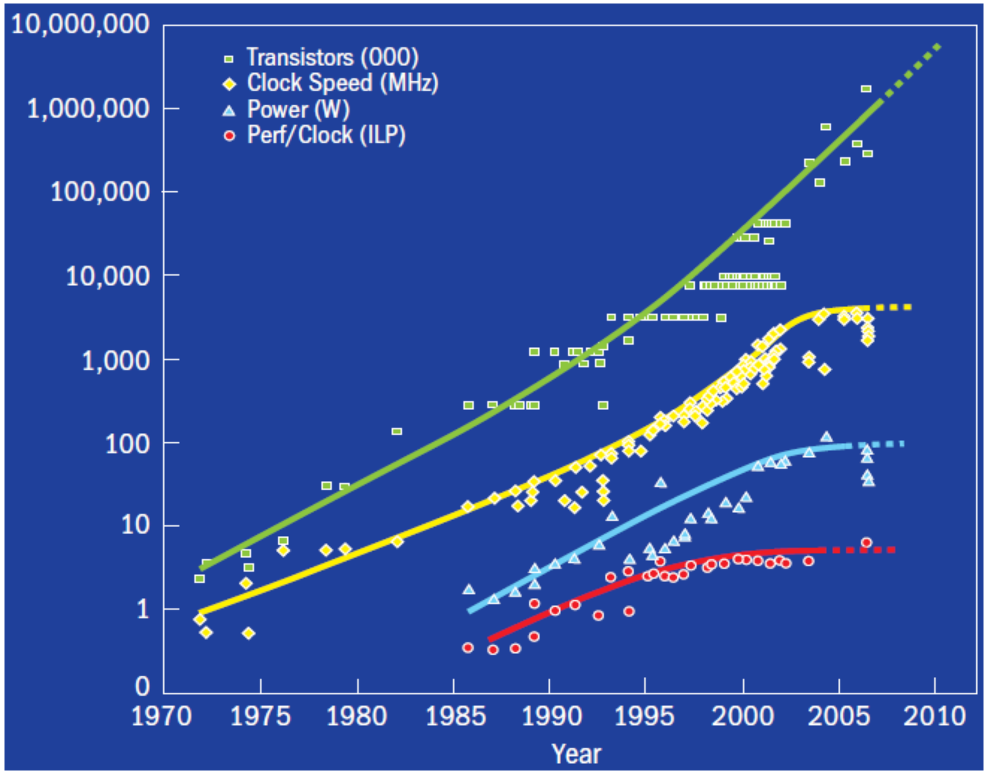
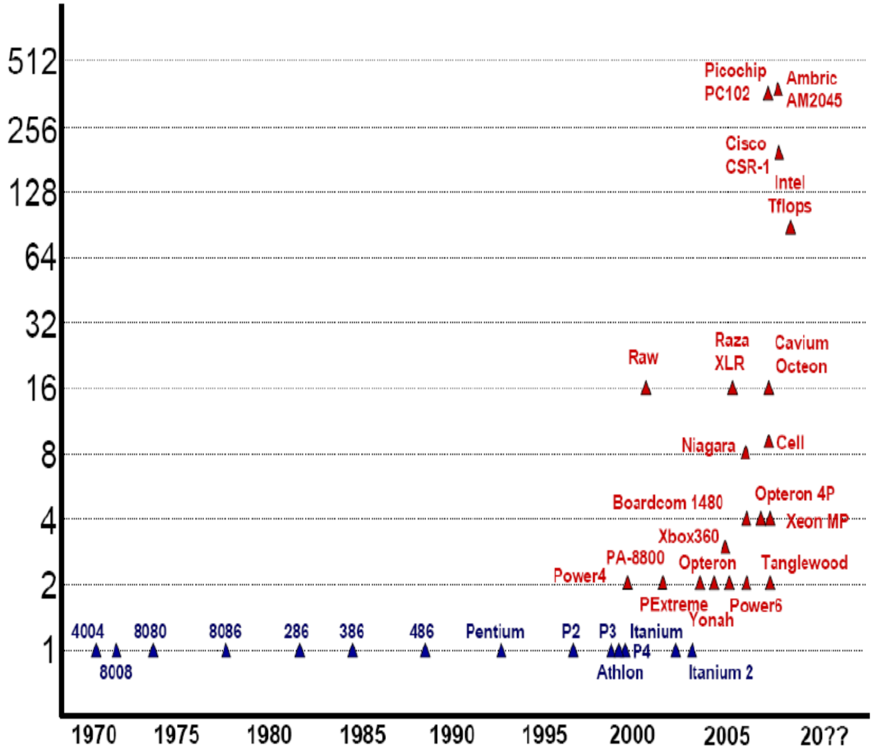


Illustration: A. TOVEY, Source: D. Paterson, UC-Berkeley



Source: Saman Amarasinghe, MIT (6.189 2007, Lec1)

Many-Cores: Major Approaches & Their Programming Environments

cc-NUMA: cache-coherent Non Uniform Memory Access

- All programming environments (paradigms) are supported
  - Shared Memory: PThreads, OpenMP, TBB
  - Message Passing: MPI
  - OpenCL
- Multi-chips: Intel Nehalem, AMD Opteron
- Single-chip: Tilera Tile-Gx, Intel MIC

GPGPU: General Purpose computation on Graphics Processing Units

- Programming environments: OpenCL, CUDA
- Need a CPU to drive the GPU
- Example: NVIDIA, AMD

MPPA: Massively Parallel Processor Array

- Programing environments: Properaty SDK (c/c++, OpenCL)
- Example: Kalray, Adapteva

How To Rate a Many-Core?

Programming effort (smaller is better)

- Compatibility is an important issue as a huge investment have been already made to develop applications, computing kernels, data bases, etc.
- Time-To-Market requires the reuse of existing software, known and standard programing languages and software technologies.

Performance (higher is better)

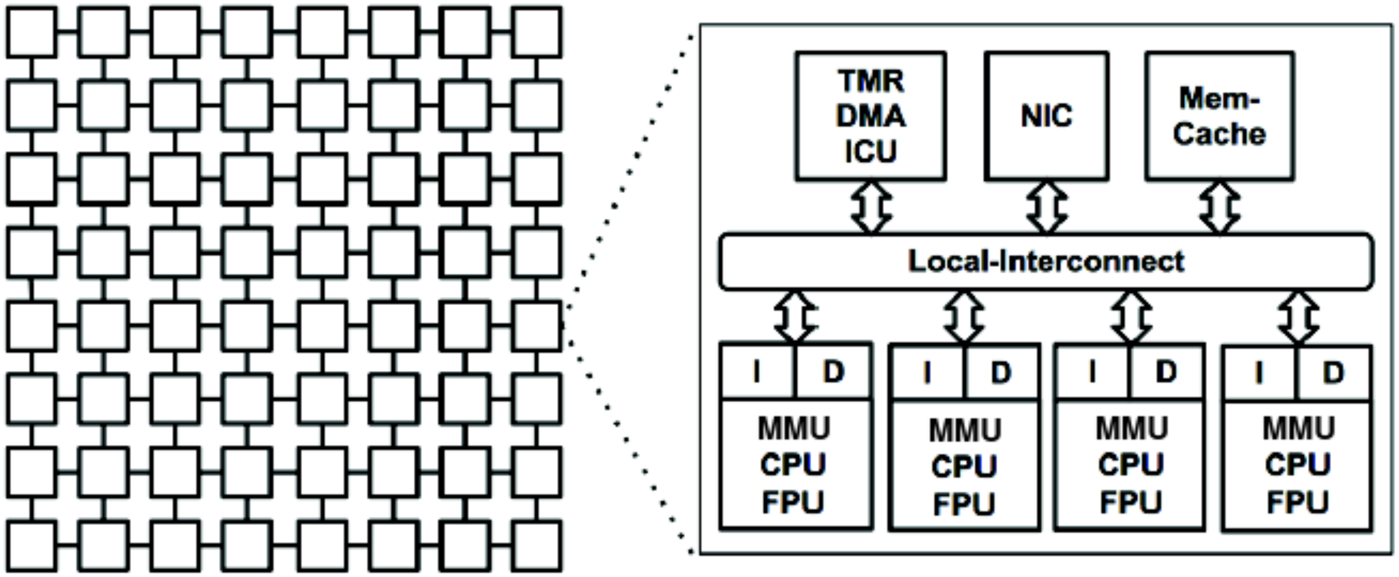
- ILP and clock-speed are no longer a source of performances
- Programs must be parallel in tasks to get performance,
- Performance depends on the scalability of hardware, OS and application.

Energy consumption (lower is better)

- Reduce the per-core area by using simple cores (static and dynamic conso.).
- Reduce the remote data access (energy by moved bit).

Single-chip cc-NUMA many-cores provide the best trade-off regarding to programming-effort and performance per-Watt

TSAR (Tera-Scale ARchitecure)



- European MEDEA+ funded project (#2A718)
- Designed by LIP6 & BULL
- **Single-chip cc-NUMA scalable up to +1024 cores**
- **Simple 32-bits RISC** cores
- DSPIN **GALS 2D mesh - NoC**
- Can run a commodity operating system like Linux & BSD

The Locality of Memory Accesses

- The locality of memory accesses is a main issue in NUMA architectures
  A poor locality leads to a performance collapsing and to a higher power consumption.

- The granularity of placement control is the physical page (MMU)

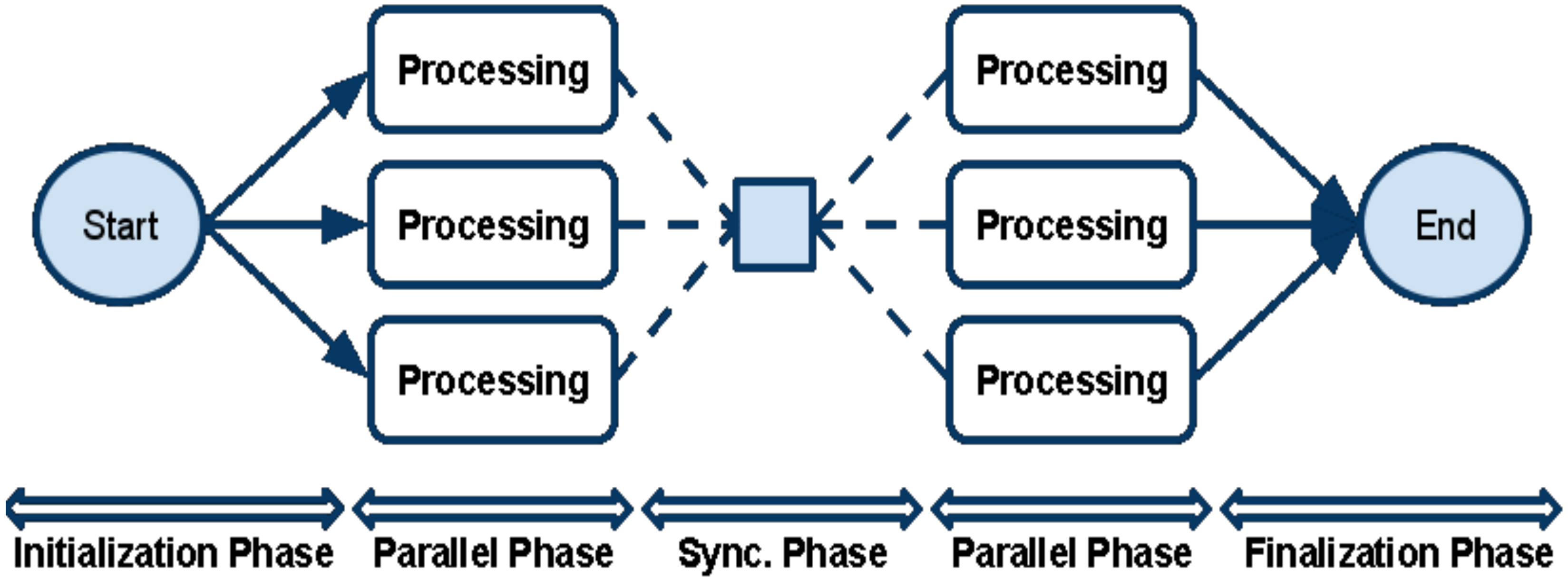
- Different techniques to enforce the locality of memory accesses of a task
  - Allocation: allocate the requested physical page locally or near to core
  - Migration: move a given page to nearby the core that most using it
  - Replication: replicate a physical page to make it local to all cores

- Two schools of thought regarding the locality handling
  - Let the programmer manage explicitly the locality (Linux)
    - Expose the hardware topology by a specialised API (NUMActrl)
    - Not standard, Not portable,
      Performances are tightly coupled with the underlying machine
  - **Let the Operating System handle transparently the locality (ALMOS)**

A Case Study for ALMOS Operating System on The TSAR Single-Chip cc-NUMA Many-Core

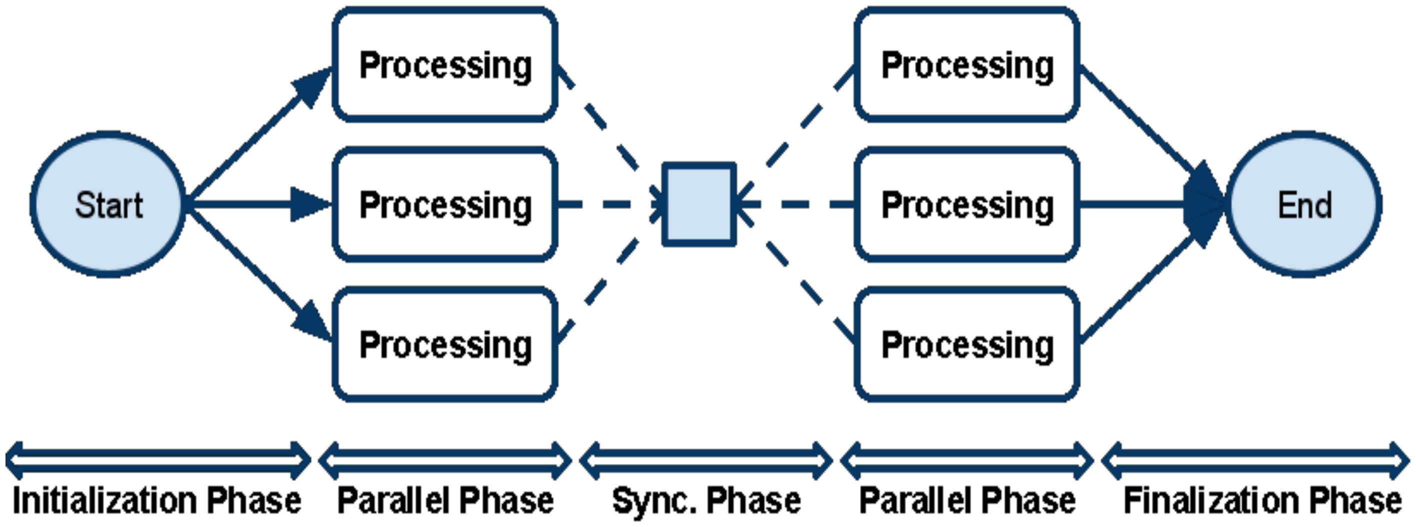
Lets see some examples where ALMOS handles the locality of memory access **transparently** to the programmer, but first, lets see the used parallel applications.

Investigated Parallel Applications



- 4 **unmodified** benchmarks: SPLASH-2 FFT, EPFilter, Histogram, Tachyon
- Two cc-NUMA Targets:
  - ALMOS / TSAR with 256-cores (CABA emulated)
  - Linux / AMD Opteron Interlagos with 64-cores

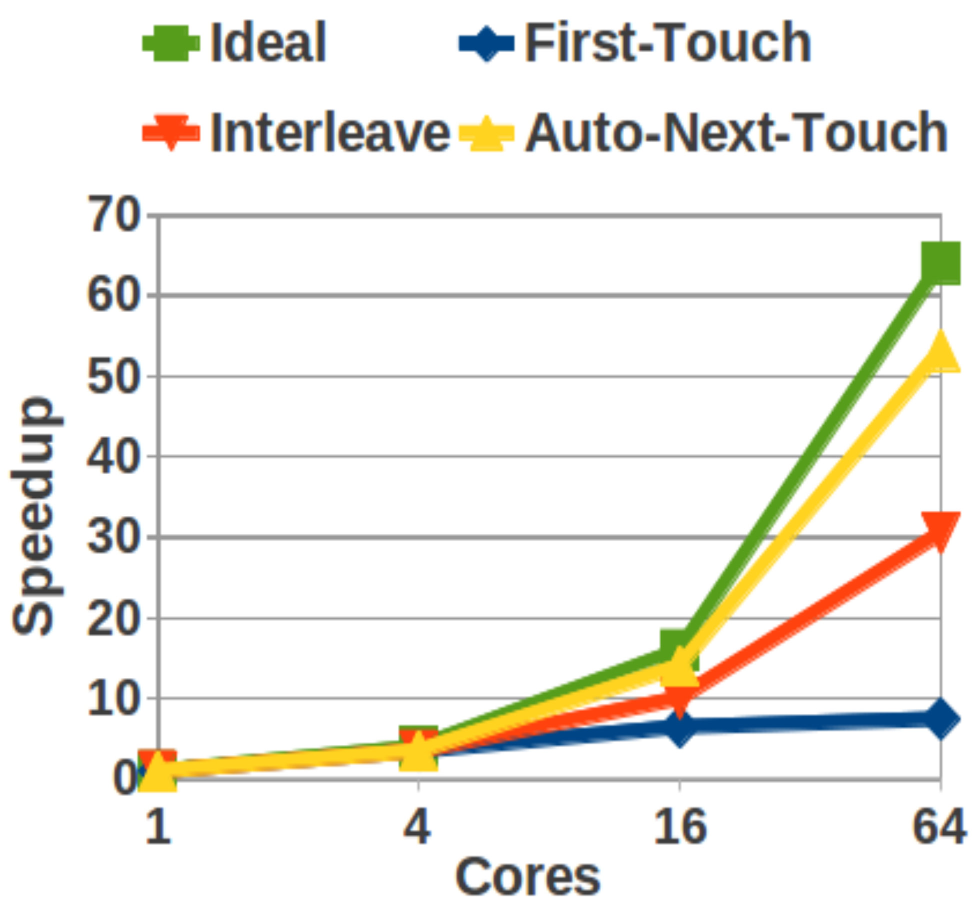
Investigated Parallel Applications

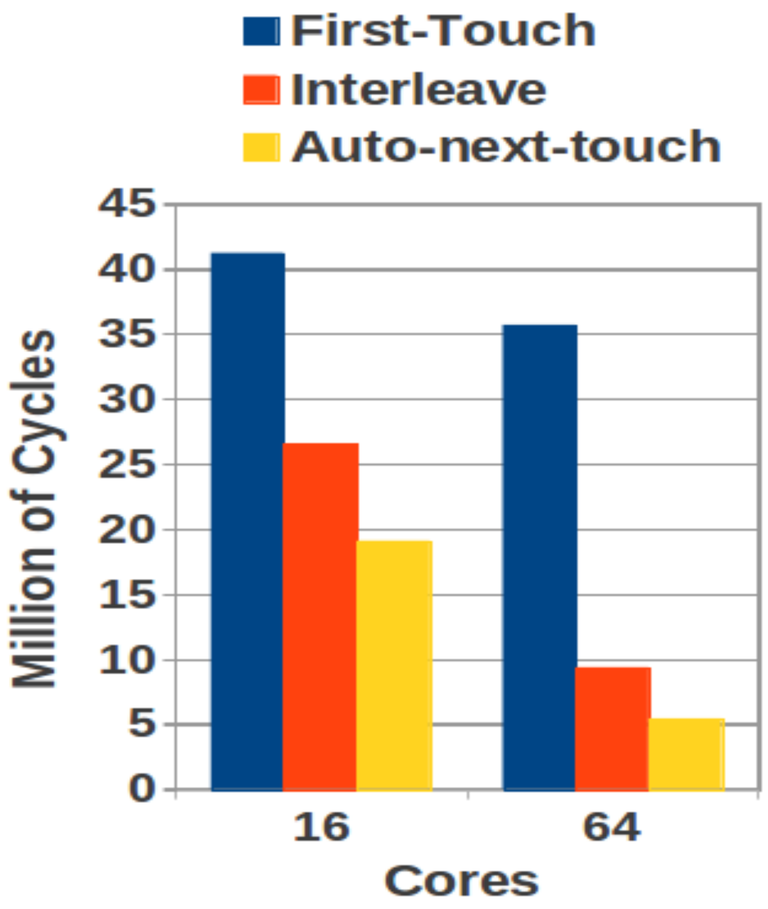


- **Unmodified** 2 HPC applications
- SPLASH-2 / FFT & Philips EPFilter (medical-images filtering)
- Highly muli-threaded single applications based on PThreads
- All-To-All inter-threads communication scheme stressing NoC & caches
- Small exécution time suitable for full-system CABA emulation

Locality of Data Access

SPLASH-2 FFT (M18) on ALMOS/TSAR





Other Locality-Related Issues

- Memory accesses made by a core include
  - L1-Data miss
  - L1-Instruction miss
  - TLBs (Data & Instruction) miss

- More threads implies the usage of more cores which implies in turn more traffic for L1-I and TLBs misses.

- The current notion of threads sharing the same process address space implies that the code and page-tables cannot be replicated.
  - Almost all accesses are remote
  - They cause a bottleneck on few memory-controllers

- ALMOS introduces kernel-level solutions to deal with code and page-tables replication in each cc-NUMA node.
  - cc-NUMA optimised implementation of PThreads
  - Native support for PGAS (Partitioned Global Address Space) paradigm

Evaluation of 4 Image & Signal Processing Parallel Applications

- Question: what kind of performances we might expect from a single-chip cc-NUMA many-core having small cores and caches?

- Metrics are:
  - Ease of programming
  - Performances per-Watt

Linux 2.6.39

AMD Interlagos 64-cores

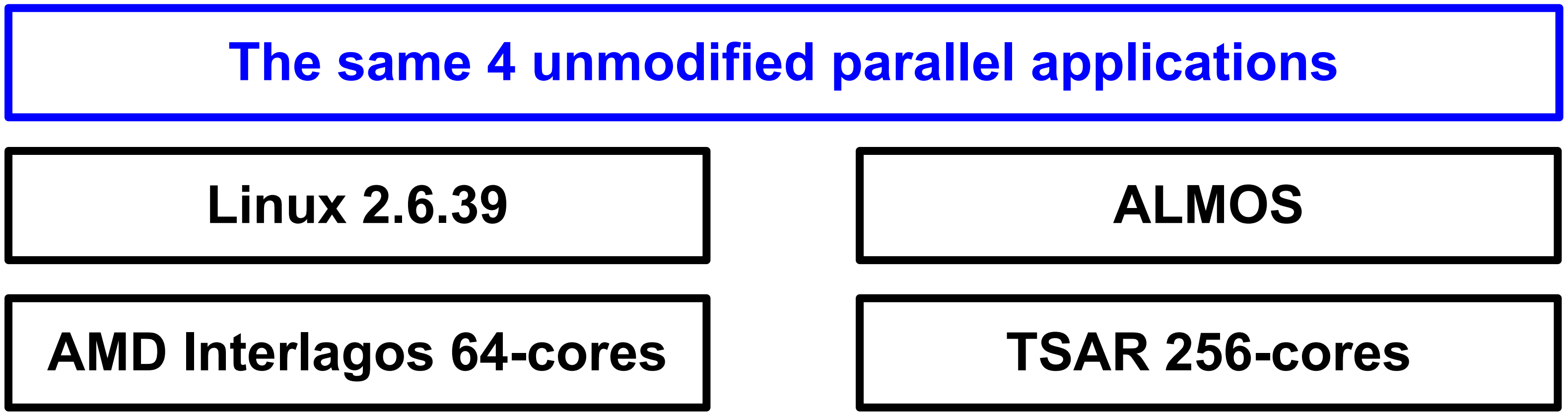
ALMOS

TSAR 256-cores

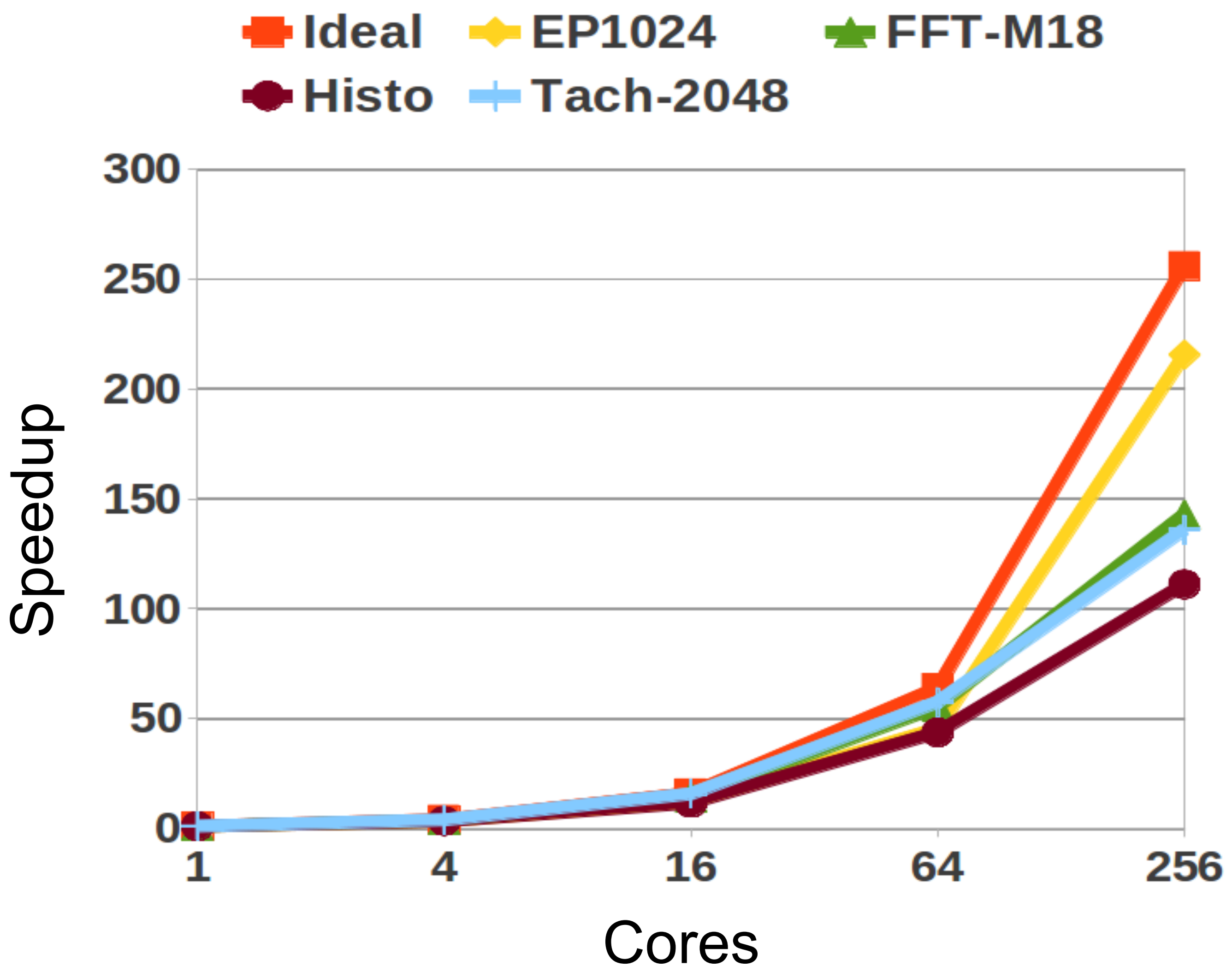
Evaluation of 4 Image & Signal Processing Parallel Applications

- Question: what kind of performances we might expect from a single-chip cc-NUMA many-core having small cores and caches?

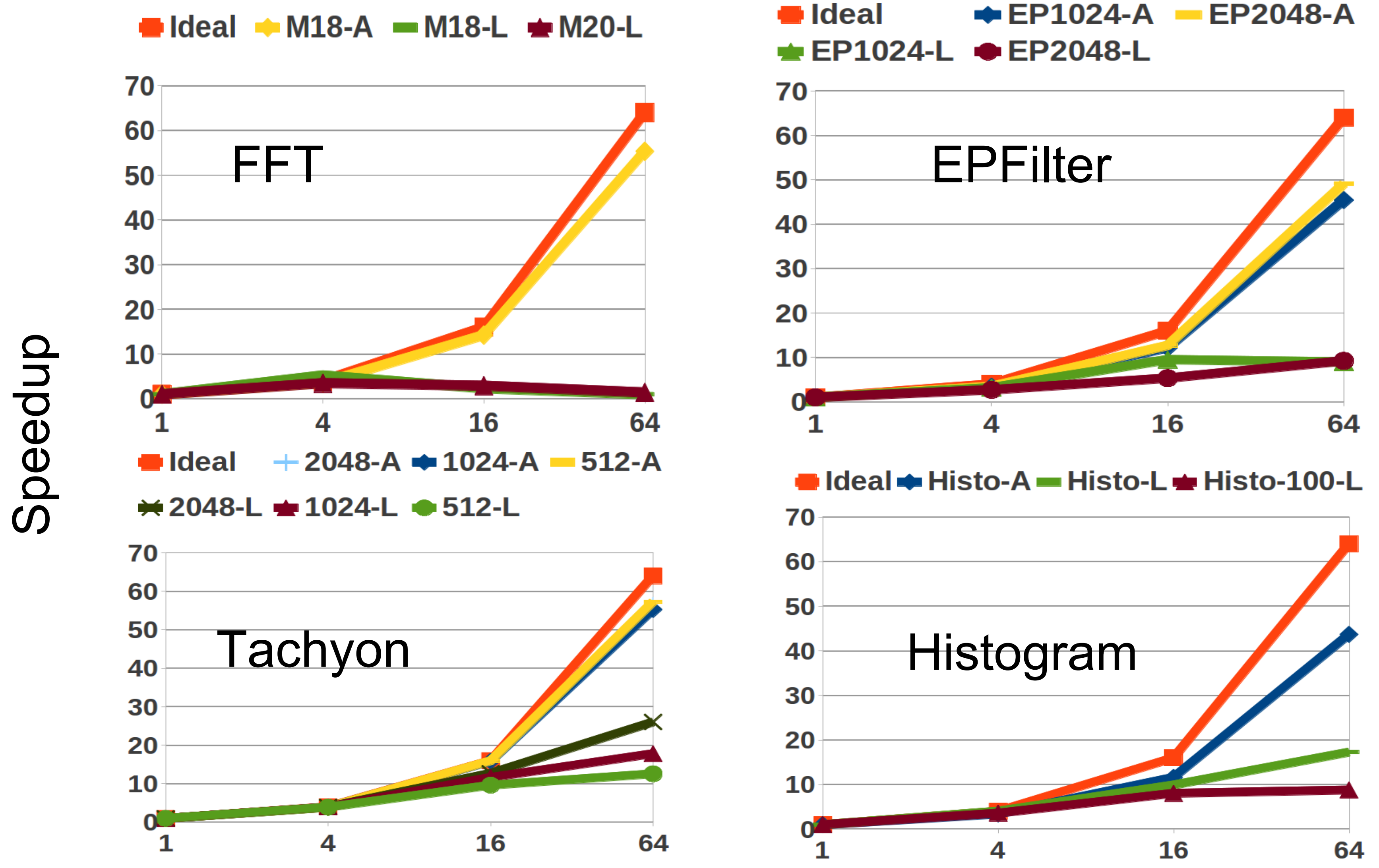
- Metrics are:
  - Ease of programming
  - Performances per-Watt



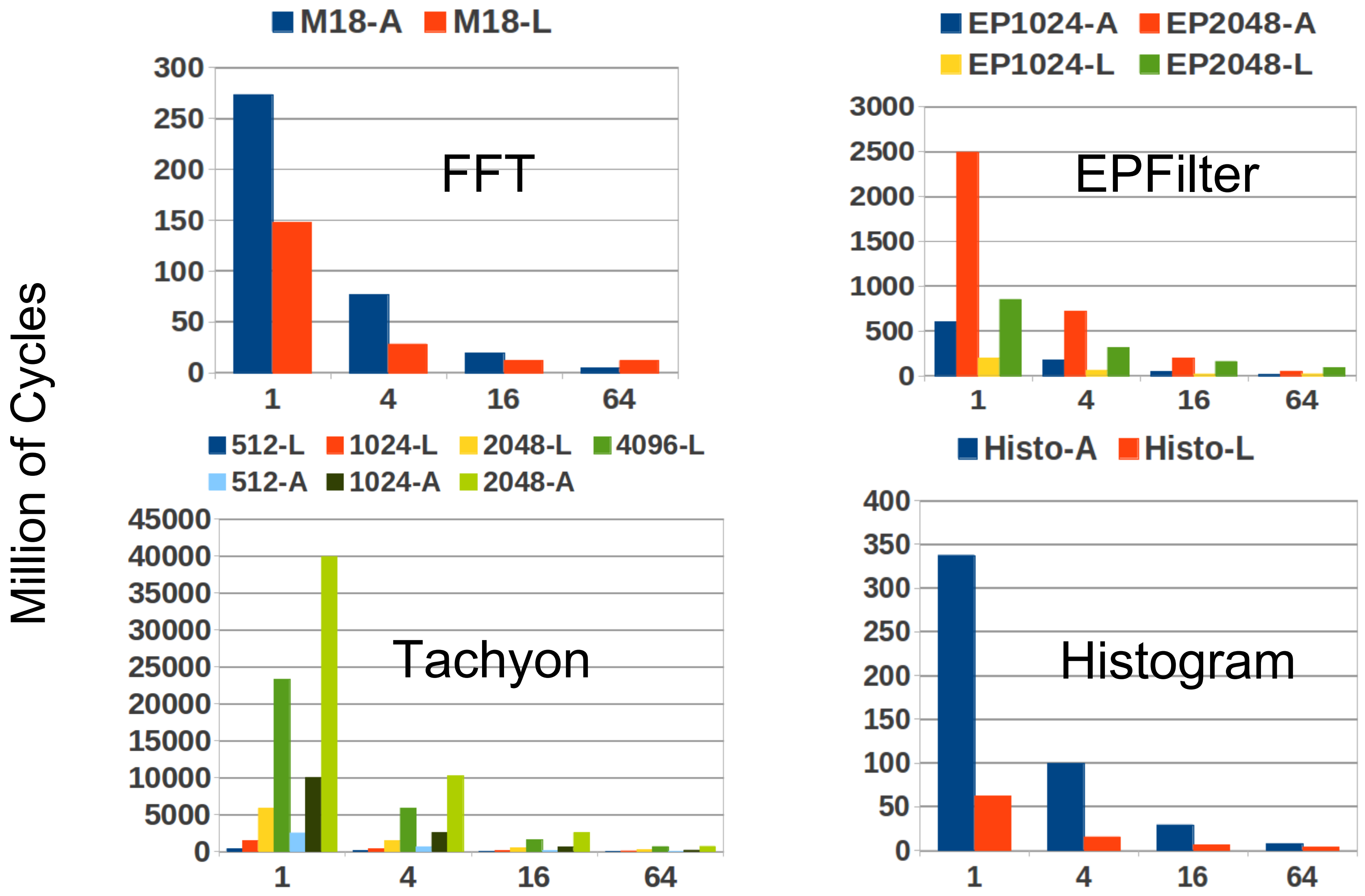
Scalability Evaluation on ALMOS/TSAR



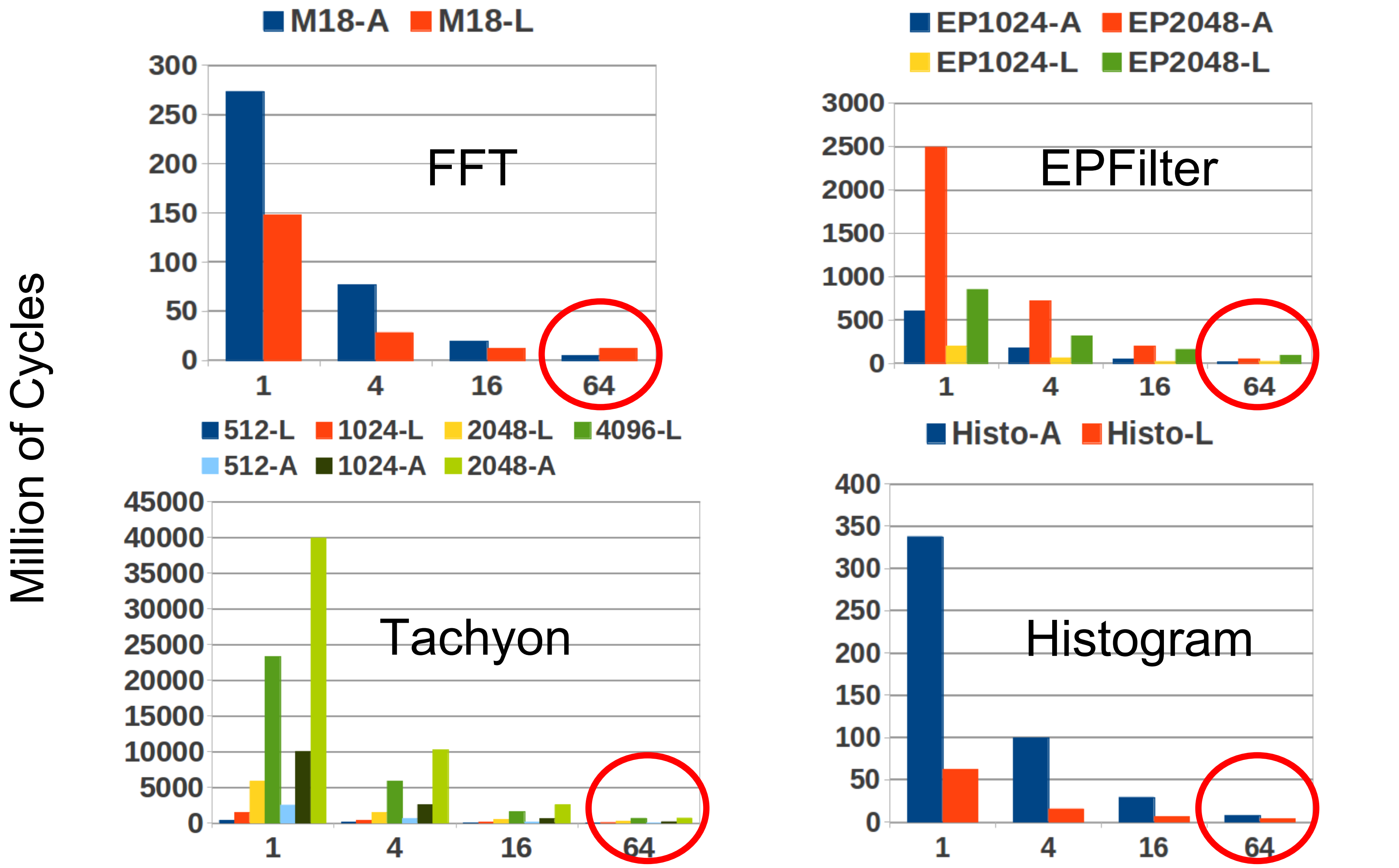
Scalability: ALMOS/TSAR & Linux/AMD



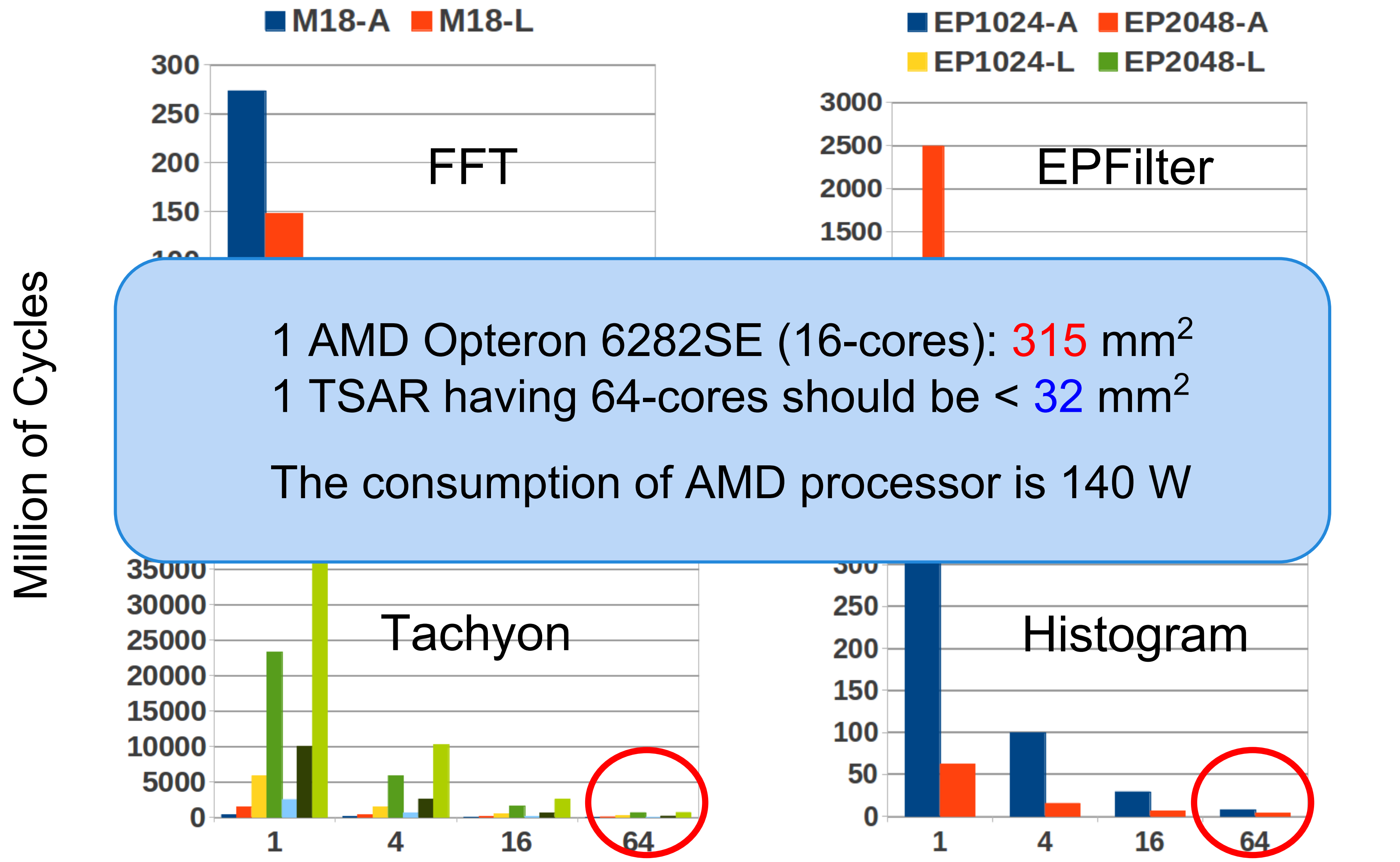
Execution Time: ALMOS/TSAR & Linux/AMD



Execution Time: ALMOS/TSAR & Linux/AMD



Execution Time: ALMOS/TSAR & Linux/AMD



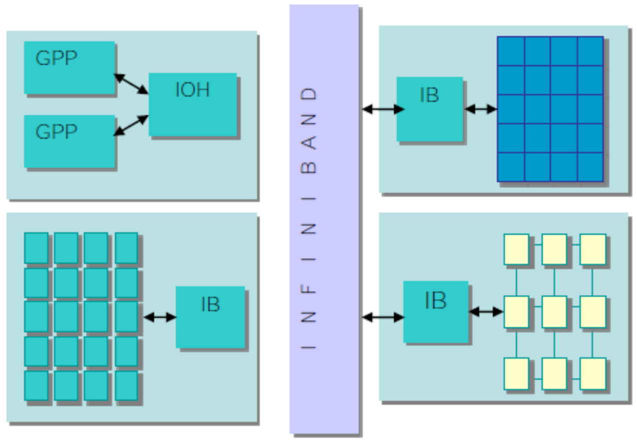
1 AMD Opteron 6282SE (16-cores): 315 mm^2
1 TSAR having 64-cores should be < 32 mm^2

The consumption of AMD processor is 140 W

TSAR/ALMOS What is Next?

SHARP: Scalable Heterogeneous ARchitecture for Processing

- Partners: BiLCEM, Bull, CEA/Leti, DVLX, Linera, METASymbiose, OPTISIS, Thales, UPMC/LIP6.
- Project leader: Bull
- Goal: developping an HPC prototype
  - High-end host CPUs
  - Several accelerators: TSAR, GPGPU, FPGA



- Our lab is working on the integration of TSAR as hardware accelerator powered by ALMOS
- A small-scale prototype of TSAR has been successfully emulated on FPGA
- We investigate the programming model to be used
  - Hybrid model (MPI/OpenMP, MPI/PThreads)
  - Unified model based on PGAS (Partitioned Global Address Space)

Conclusion

- The number of cores is the new indicator instead of processor's frequency
- Different industries use/need many-cores with high demand in the near-future
  - Embedded systems
  - High-End computers
  - HPC (High Performances Computing)
- Single-chip cc-NUMA many-cores provide the best trade off for programing-effort, performance and power-consumption.
- Our lab proposes an efficient single-chip cc-NUMA many-core architecture, TSAR, with an optimized execution environment, ALMOS.
- Our lab is working on several enhancements for both of TSAR architecture and ALMOS (e.g. cache-hierarchy optimisation controlled by the kernel, 3D-L3).
- We are seeking for further cooperations, **are you interested?**