



# **Microarchitecture performance assessment and energy monitoring of MaX codes through Linux Perf and power management API interfaces**

**HPCSE 2022**, Hotel Soláň, Czech Republic  
May 16-19, 2022

**Daniele Cesarini**

HPC Specialist - CINECA

Federico Ficarelli  
Federico Tesser  
Andrea Piserchia  
Fabio Affinito

# Overview

- *Introduction on HPC microarchitectures*
- *Performance and efficiency of microarchitectures*
- *Jump into HW performance counters*
- *A view on Linux Perf*
- *How to monitor energy/power*
- *Roofline and TMAM performance models*
- *Experimental results on MAX codes*
- *Conclusions & Take away messages*

# Acknowledgments



## MaX Project

Most of the experimental work presented in these slides was performed in the WP4 codesign work package of MaX project



## Regale Project

The software stack and the power management tools was developed in the REGALE project

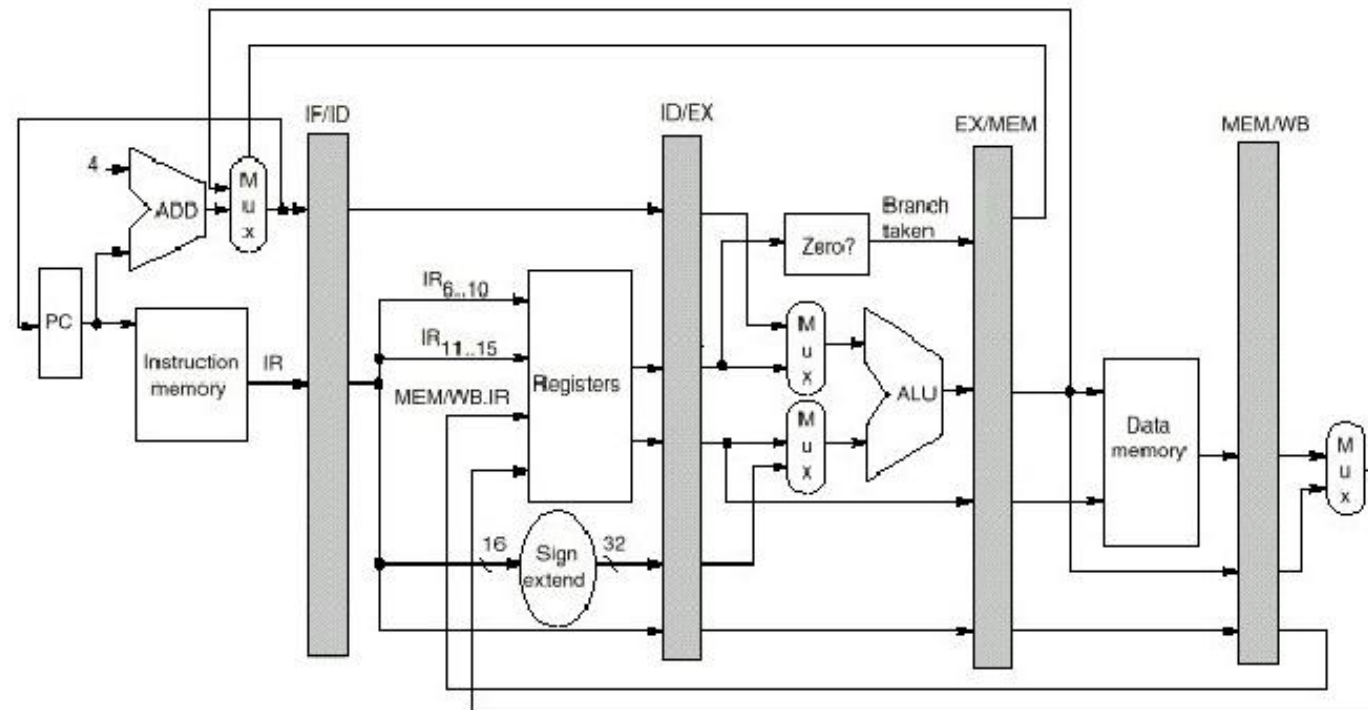


## EPI Project

The microarchitecture analysis and the performance assessment are part of the work of EPI-SGA1/2 projects

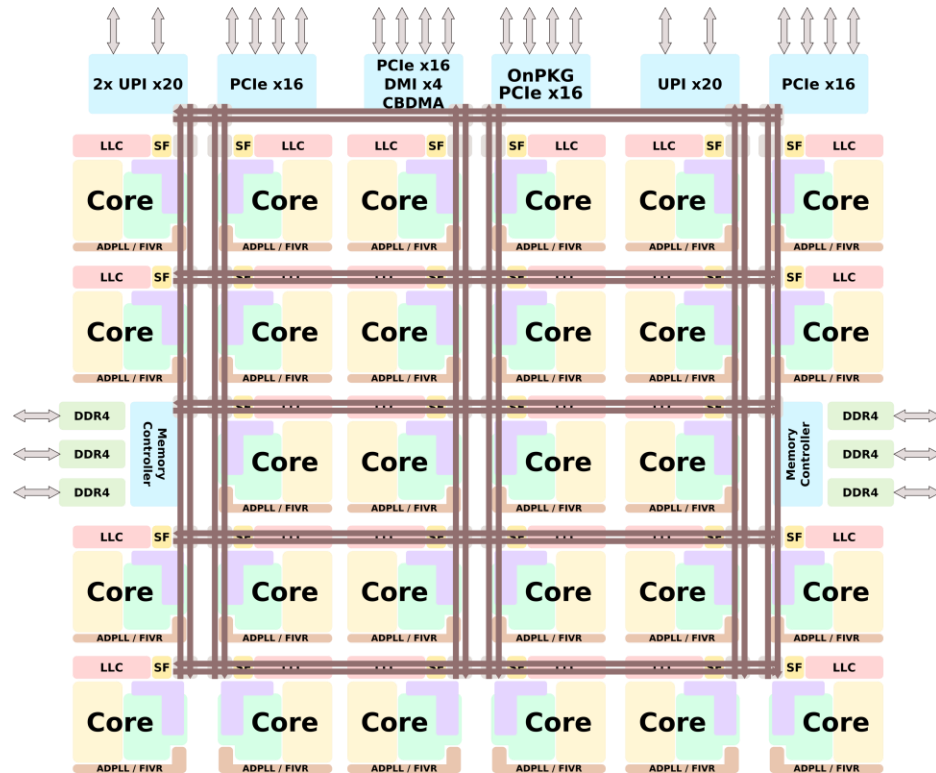
# What they taught to you about *computer architecture*

**CPU:** scalar, in order, RISC based, 32bit, short pipeling, single level cache, single threading, single core, single socket with fixed operating frequency and uniform memory access



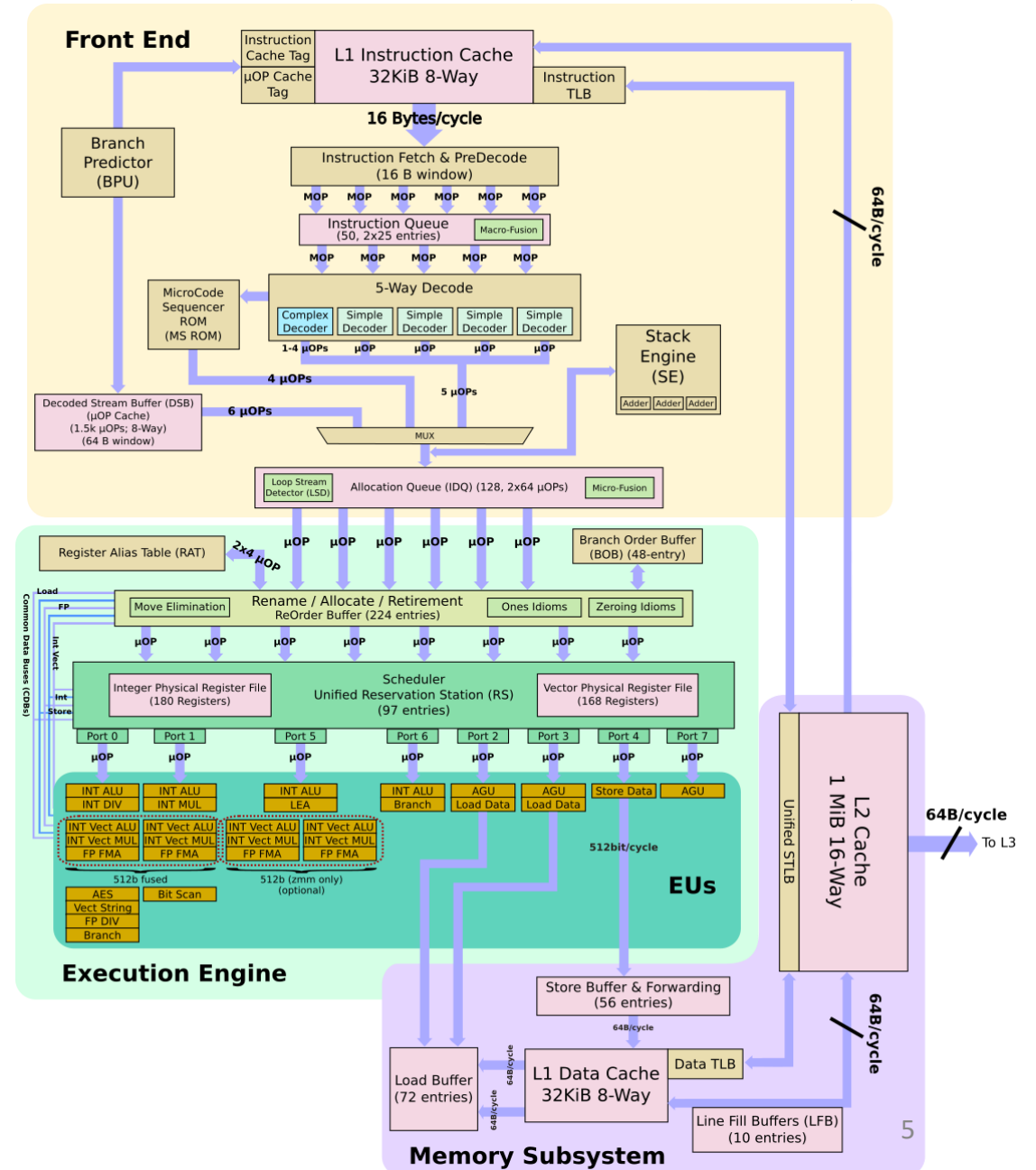
# What it actually is

**CPU:** superscalar, out of order, multi-level caches, CISC based (x86), 64bit, multi-threading, many core, multi socket with dynamic voltage and frequency scaling and non-uniform memory access



Cascade lake SoC Layout

## Cascade lake microarchitecture




# Goal: exploit performance

## Performance is a result of:

- How many instructions you require to implement an algorithm
- How efficiently those instructions are executed on a CPU

## But what does it mean "efficient execution"?

- **Scientific view:** HPC application → Scientific algorithm + data → Result
- **Computer view:** HPC application → Set of finite sequences of computer instructions + digital data → Result
- Computer performance → Higher Instructions Per second/Cycle (IPC) → Shorter execution time
- Almost true -> Eg. higher IPC with scalar instructions
- Floating point Operations Per Second (FLOPS) → Better metric → 

## But remember the following three things:

1. It is impossible to reach the theoretical peak performance of a system;
2. Focusing on the optimization of a single performance metric can reduce other performance metrics (trade-off problem);
3. A single performance metric cannot express the overall efficiency of a microarchitecture but we need to consider multiple metrics;

# *Micro architecture performance optimization*

In modern CPUs it is very difficult to understand if my application is efficiently performing on a specific system (also from an energy point of view)!

We need HW support from the processor (PMU) -> **Only what is measurable can be improved!**

Tools help you to get access to the HW subsystem and automatize routines but...

**Use your brain! Tools may help, but you do the thinking!!!**

## *μarch performance events (very few of them)*

- **Cycles:** count the number of cycles
- **Instructions retired:** count the number of macro instructions executed
- **Vector instructions retired:** count the number of vector macro instructions
- **Branches:** count the number of branch taken
- **Cache miss/hit** (at multiple cache levels): count the miss/hit of the cache references -> it show the locality of the code
- **Memory read/write:** number of time that a cache line was read/written from the memory

## *μarch performance events ≠ performance metrics*

- **FLOPS:** arithmetic operations executed
- **Memory throughput:** number of bytes exchange with the memory
- **IPC:** instructions per cycle -> this metric show the macro instruction throughput of the microarchitecture
- **Vectorization ratio:** percentage of how many vector instructions are retired wrt the total instructions



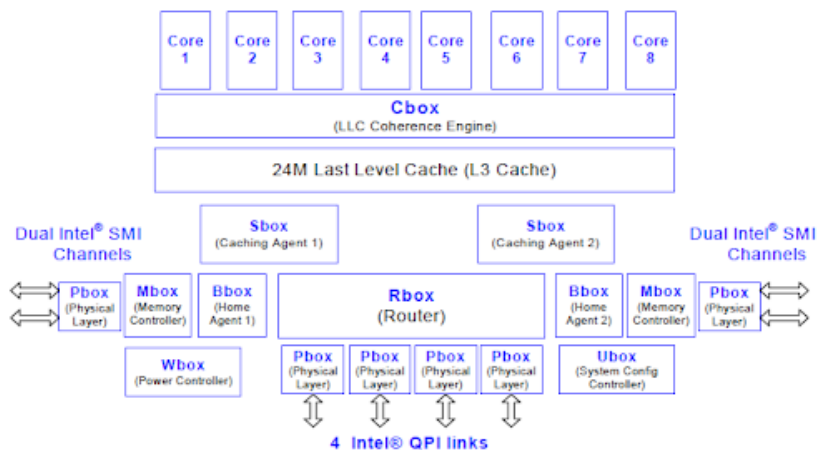
# Performance Monitoring Unit (PMU)

The CPU supports you with the PMUs!

A PMU usually support many events (cycles, instructions retired, etc.) through **Performance Monitoring Counters (PMC)**.

A PMU can be:

- **on-core**: microarchitecture events at the core level (cycles, instructions retired, ...)
- **off-core**: microarchitecture events outside of cores (memory read/write, ...)



PMCs can be:

- **fixed**: can be only enabled or disabled and profile a specific event
- **configurable**: can monitor many events

PMCs are usually configurable only at kernel level

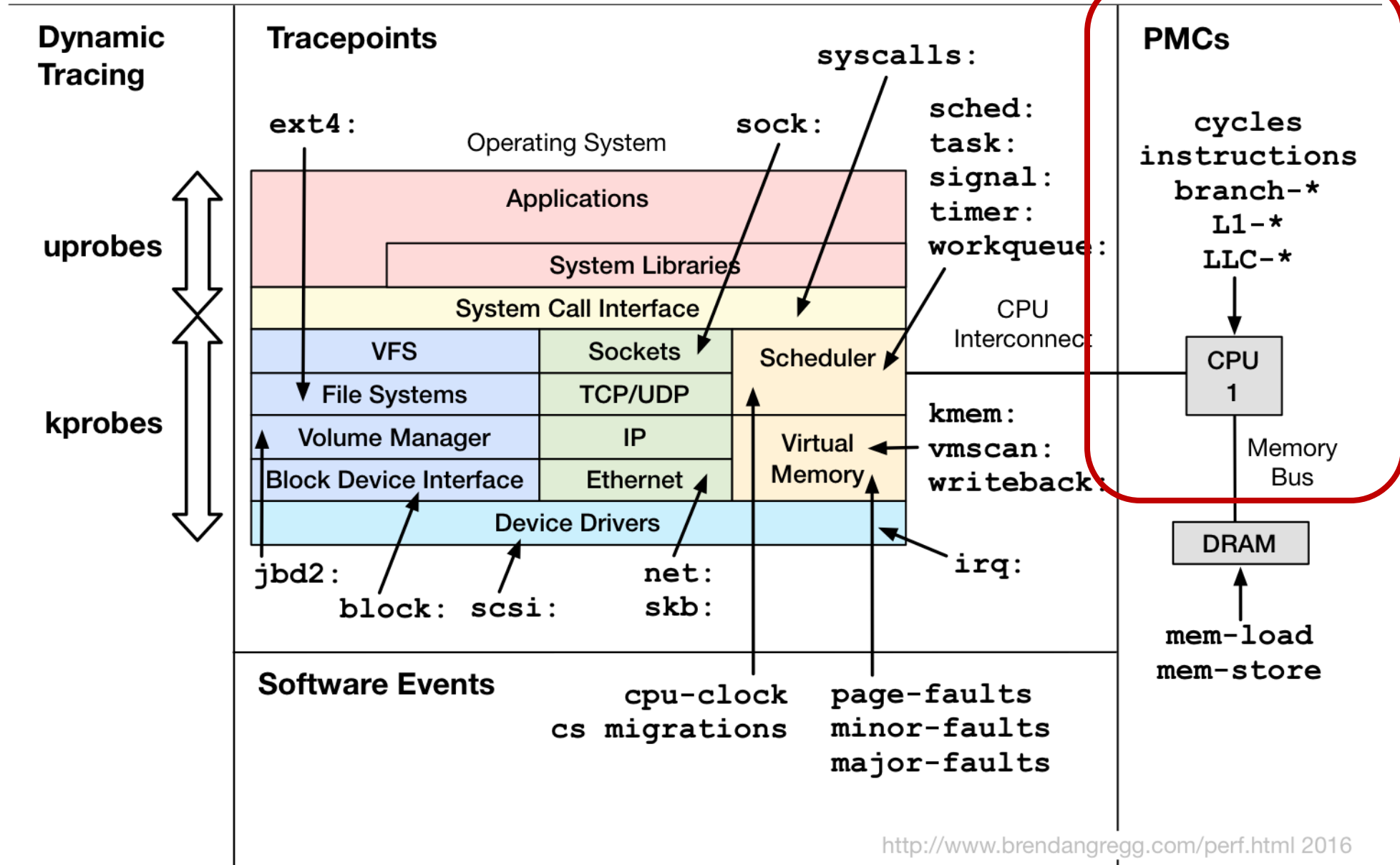
Usually, CPUs provide at user space some assembly instructions with low overhead (see `rdpmc()`) to read PMCs

# Enter *Linux perf*

- *Official* Linux profiler
  - Built on top of kernel infrastructure (ftrace)
  - Source and docs in kernel tree
- Provides a plethora of profiling/tracing features at all system levels
  - user, kernel, CGROUP, etc...
- Most important for us: **a comprehensive toolbox to gain workload execution insights via PMCs**
- Low overhead\*
  - Tunable
  - 1-2% counting mode, **5-15% sampling w/multiplexing**

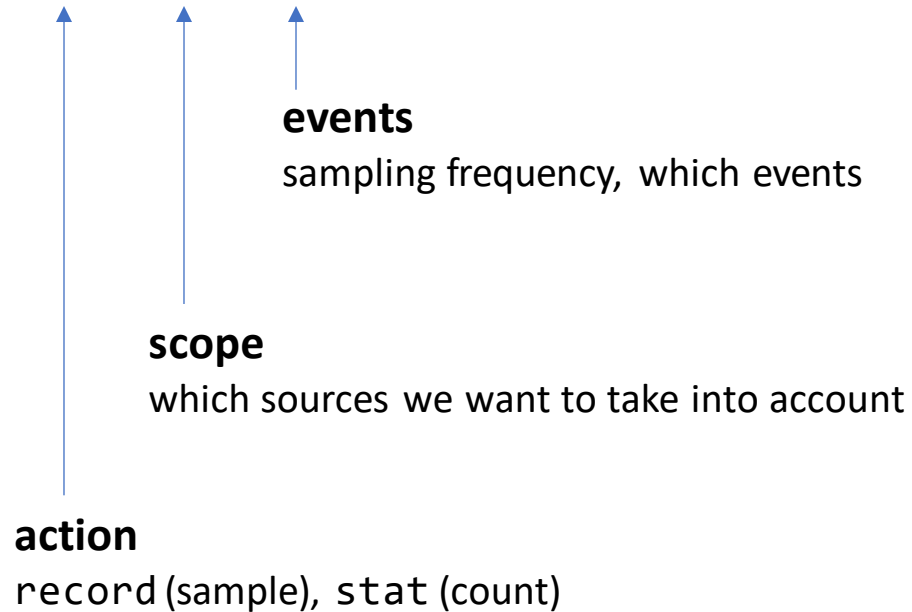
\* Nowak, Andrzej et al. "Establishing a Base of Trust with Performance Counters for Enterprise Workloads." *USENIX Annual Technical Conference* (2015).

## Linux perf\_events Event Sources



# perf stat/record format

```
$ perf record -a -F 4000 -e L1-dcache-load-misses,L1-dcache-loads -- $APP
```



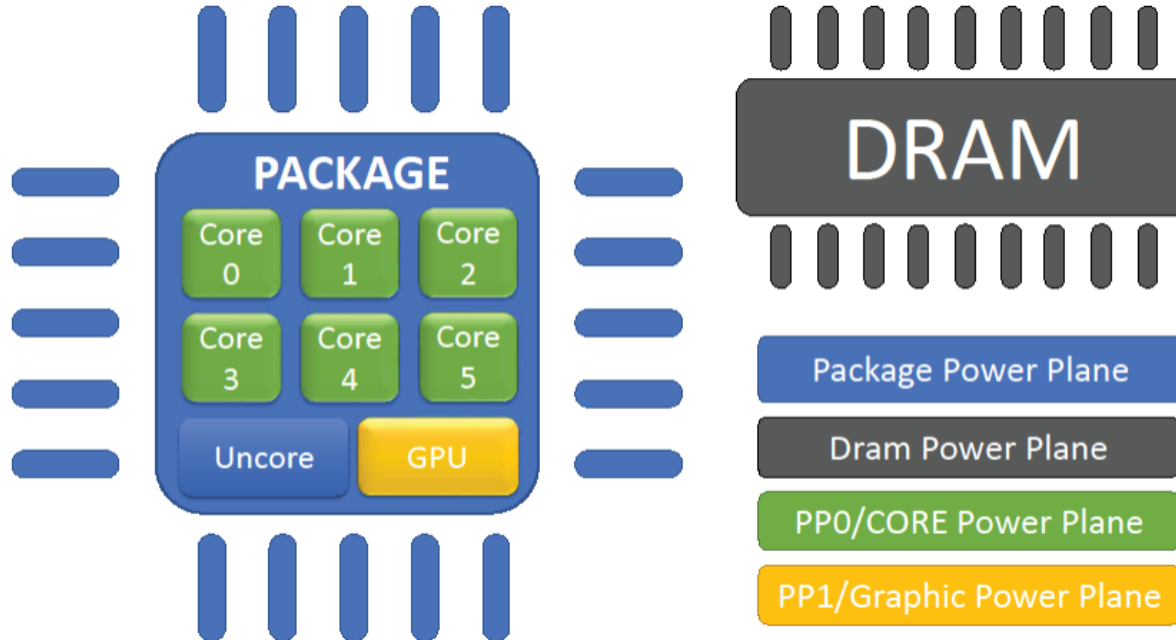
# What about *energy* and *power*?

- *We are very unlucky!*
- *No standard mechanisms/APIs/performance counters*
- Different system = different power/energy monitors
- Often only root can have access
- Several power domains
- Metric to measure energy efficiency: ***FLOPS/W***

# Intel power management - RAPL

Intel CPU implements hardware power controller called Running Average Power Limit (RAPL)

## Power Domains



**Package Domain:** monitor/limit the power consumption for the entire package of the CPU, this includes cores and uncore components.

**DRAM Domain:** monitor/limit power consumption of the DRAM memory. It is available only for server architectures. (no client)

**PP0/Core Domain:** is used to monitor and limit the only to the cores of the CPU.

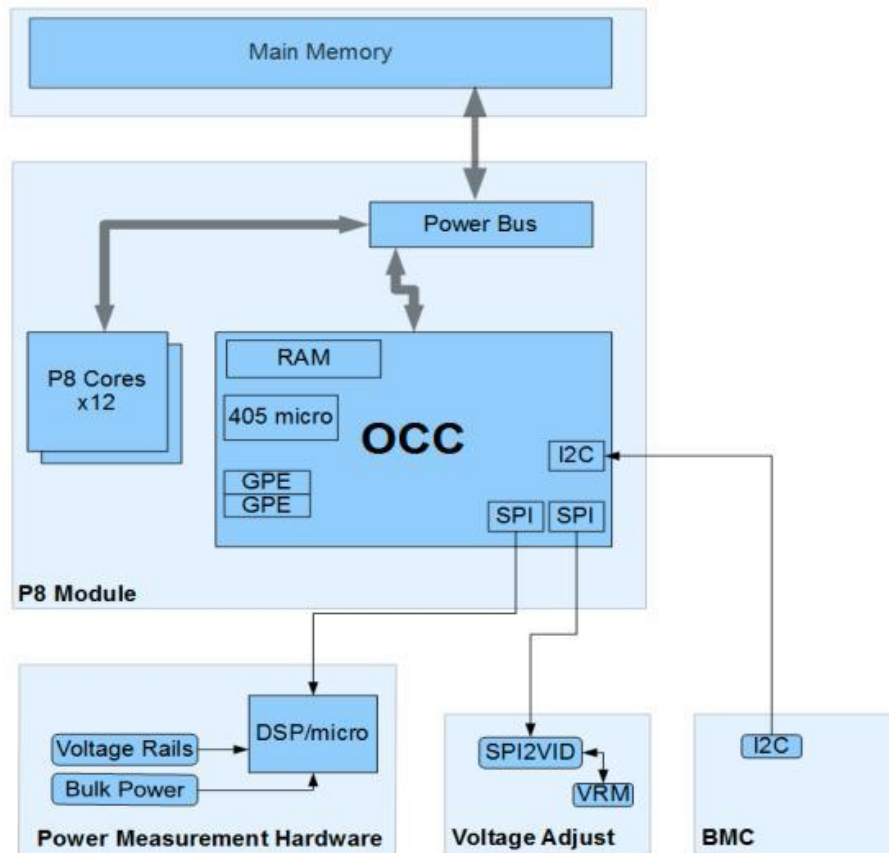
**PP1/Graphic Domain:** is used to monitor and limit the power only the graphic component of the CPU (no server).



**Machine Specific Registers:** MSR address space 0x600 – 0x640

**Sysfs Interface:** `/sys/devices/virtual/powercap/intel-rapl/intel-rapl:X/intel-rapl:0:Y` X=socket\_id, Y=power\_domain

# IBM Power8/9 Power Management - OCC



## What is the On-Chip Controller (OCC)?

- 405 microcontroller with 512k dedicated RAM
- Hardware/Firmware that controls power, performance & thermal (independent micro OS)
- Can communicate in band with the host through the main memory

## What does OCC do?

- Reads/controls system power (CPUs, GPUs, board, etc.)
- Reads/controls chip temps (CPUs, GPUs, board, etc.)
- Enables efficient fan control
- Provides thermal protection
- Power Capping
- Fault Tolerance
- Energy saving
- Performance boost



**In-band:** through Linux *occ-hwmon* kernel module: `/sys/firmware/opal/exports/occ_inband_sensors` + C data structs that define the 405 memory address space (<https://www.kernel.org/doc/html/v5.9/hwmon/occ.html>, <https://github.com/open-power/occ>)

**Out-of-band:** through the BMC and IP network communication

# Cavium ThunderX2 Power Management - TX2MON

Cavium ThunderX2 processor implement a similar RAPL/OCC controller

Unfortunately, we don't have too much information on it, but we know that is possible to monitor the following power domains:

- **Core:** voltage and power consumed by all cores on the SoC.
- **SRAM:** voltage and power consumed by internal SRAM on the SoC.
- **Internal memory:** voltage and power consumed by the LLC ring on the SoC.
- **SoC:** voltage and power consumed by miscellaneous SoC blocks.



**In-band:** through Linux *TX2MON* kernel module: `/sys/devices/platform/tx2mon` + C data structs that define the controller memory address space (<https://github.com/Marvell-SPBU/tx2mon>)

**Out-of-band:** through the BMC and IP network communication

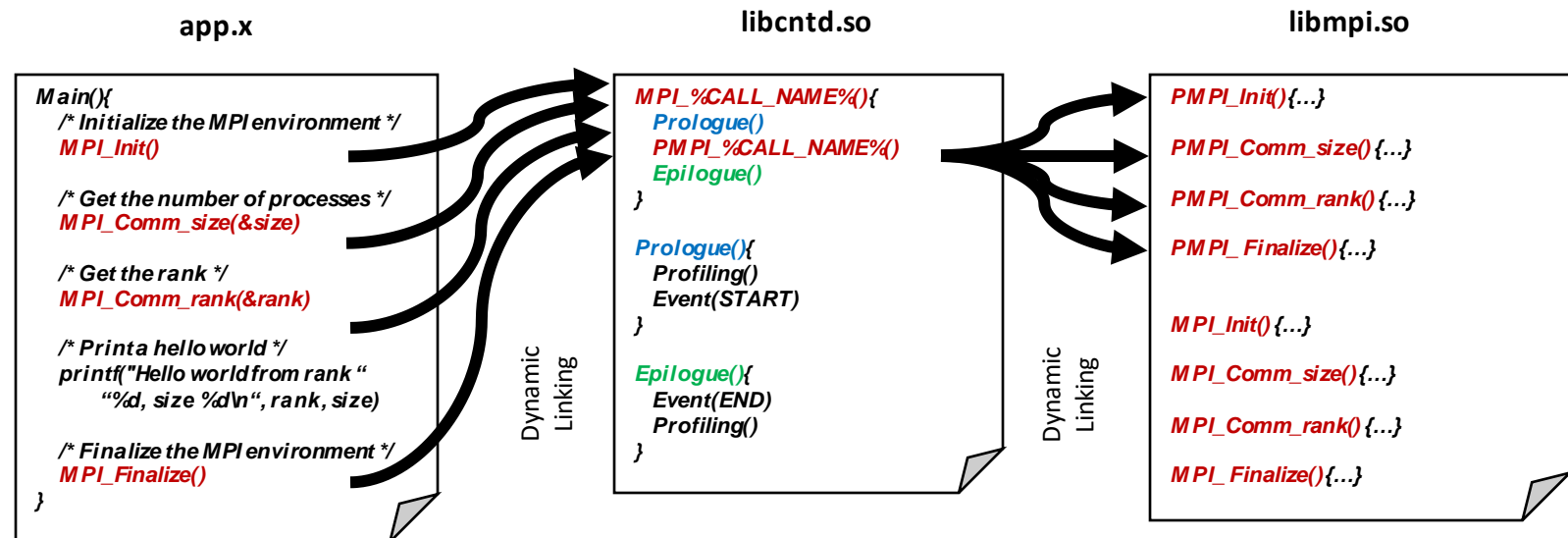
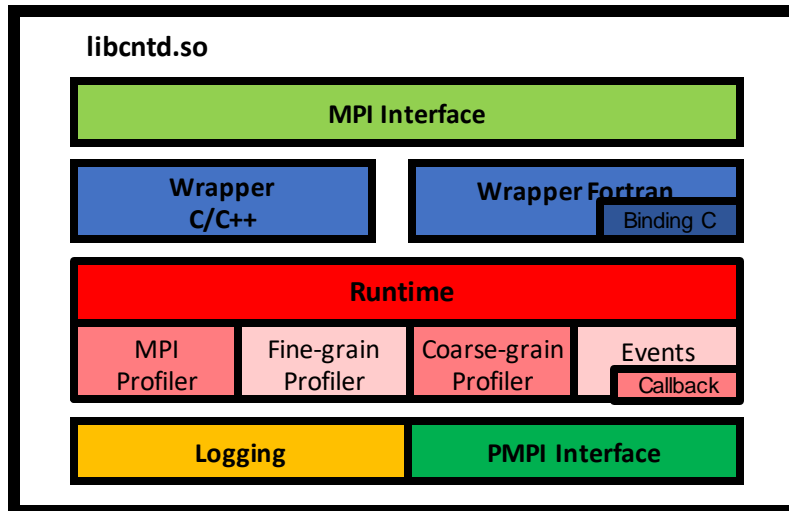


## So, how can we collect power/energy information from different systems?

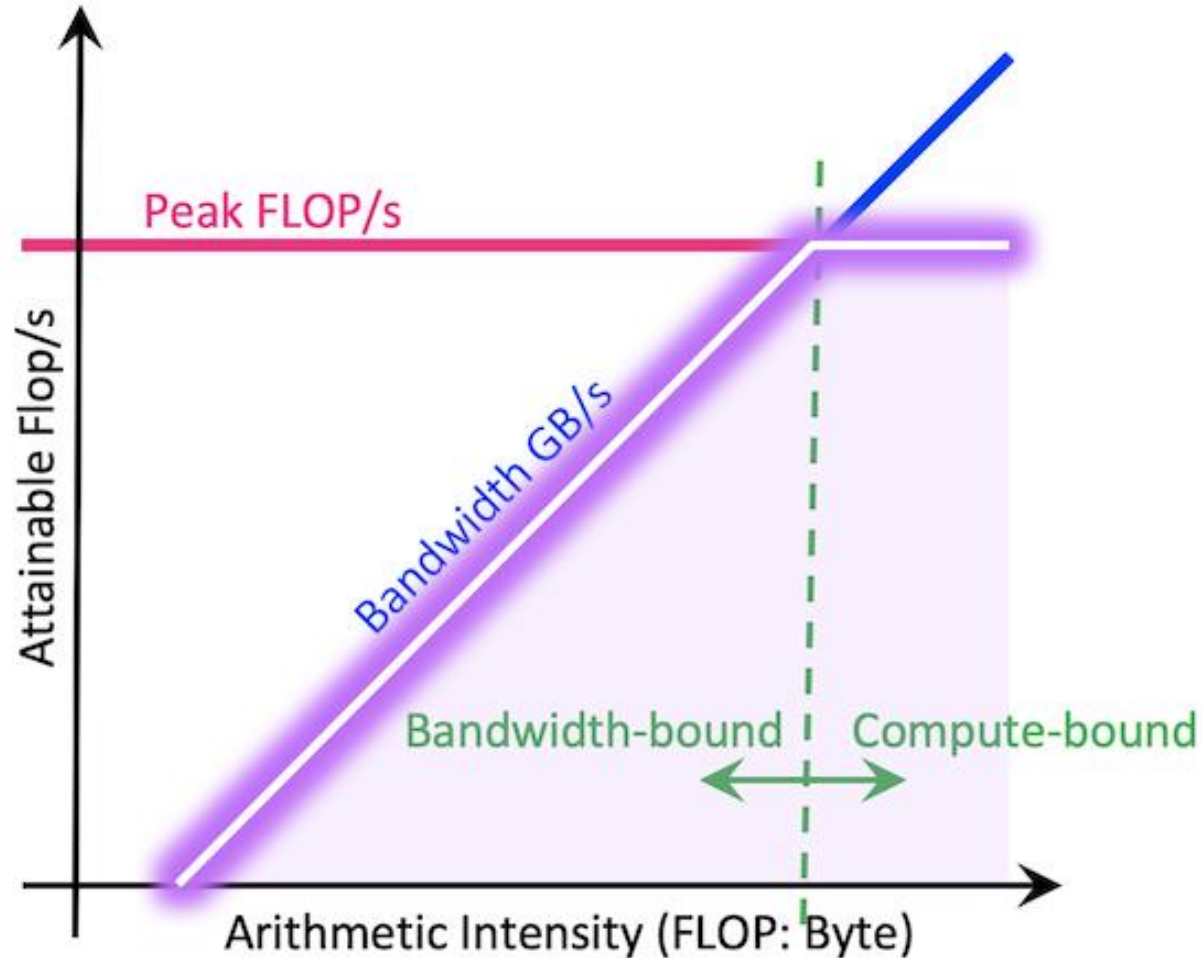
- Not easy at all!
- For this reason, we developed **COUNTDOWN** (CINECA + UNIBO project)
- **COUNTDOWN** instruments the application at execution time and collect information on the application workload and on the energy/power consumption of the system



ALMA MATER STUDIORUM  
UNIVERSITÀ DI BOLOGNA



# How can I use perf metrics? *The roofline model*

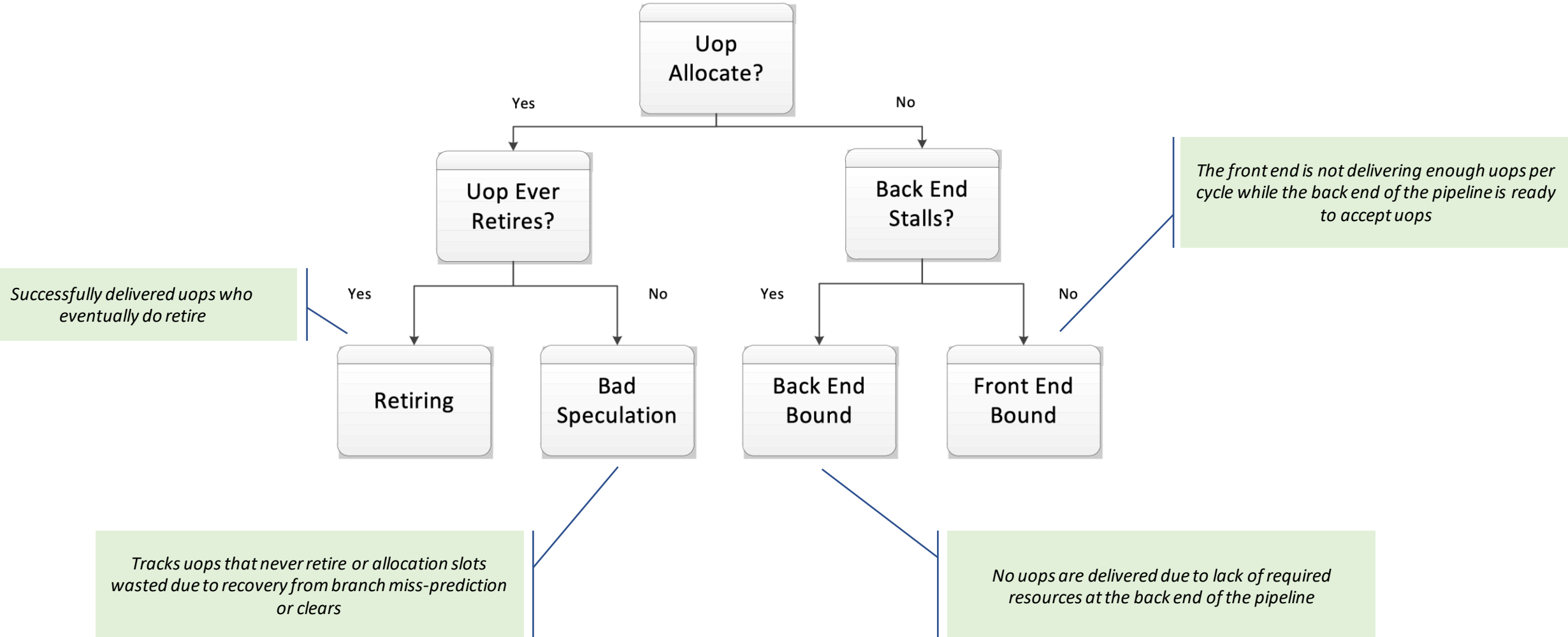


- Is performance limited by compute or data movement?
- **Y-axis:** performance in FLOPS
- **X-axis:** Arithmetic Intensity AI (FLOP/Byte)
  - Ratio between total FLOP and total byte exchange with main memory
  - Measure of data locality (data reuse)
  - Typical machine balance is 5-10 AI
  - Stream TRIAD: 0.083 AI (2 FLOP per 24 bytes)
- Application/kernel near the roofline are making **good use** of computational resources
  - Compute bound: >50% of peak performance
  - Bandwidth bound: >50% of Stream Triad
- Bad performance:
  - Insufficient cache bandwidth
  - Bad data locality
  - Integer heavy code
  - Lack of FMA
  - Lack vectorization

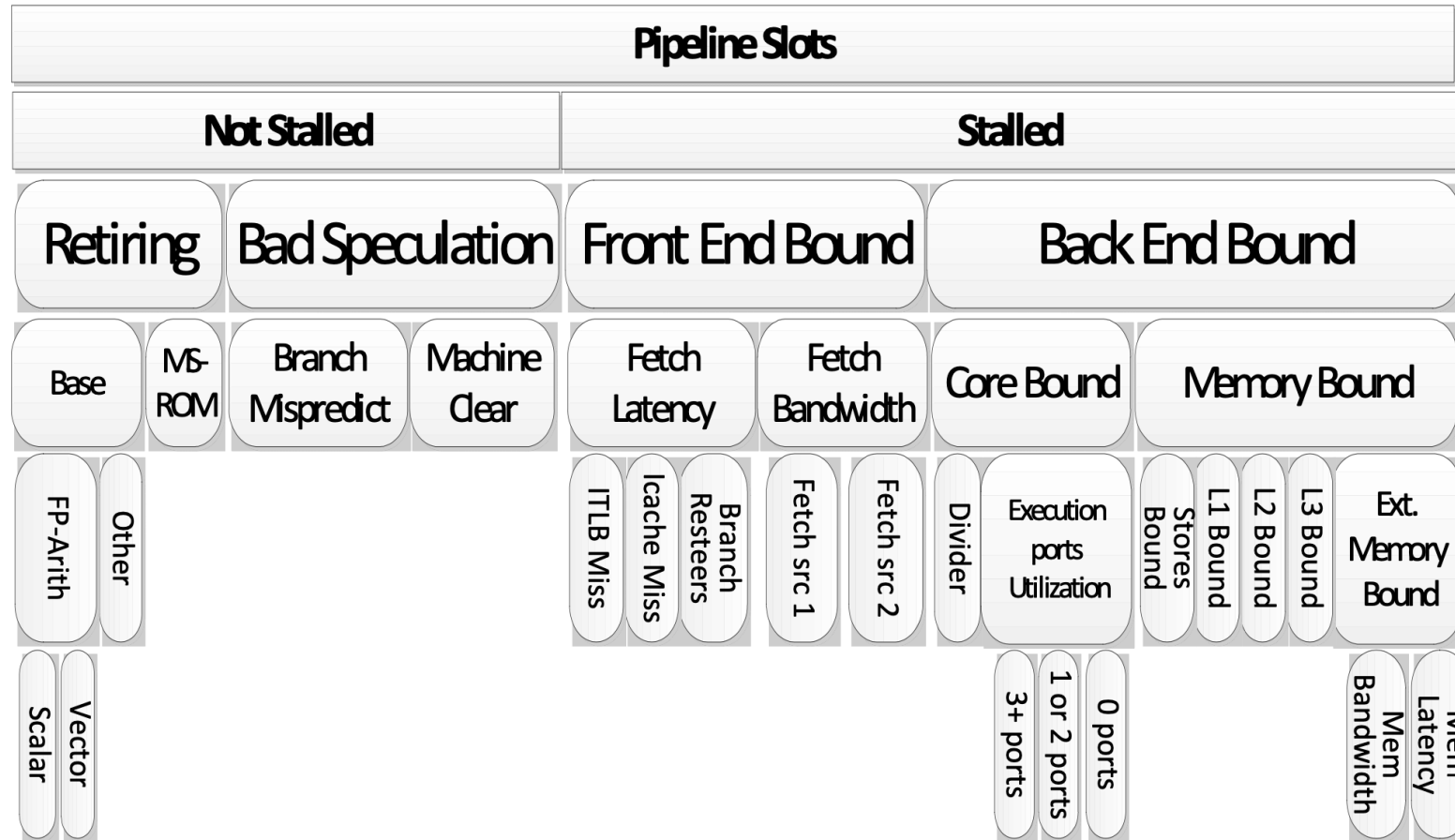
# Drill down: identify bottlenecks with TMAM

- Top-down **Microarchitecture Analysis Method**
- first attempt by Intel at a simplified approach
- **hierarchical drill-down method guided by PMCs**
- **goal: identify the real bottleneck WRT micro-architecture**

# TMAM: high level breakdown



# TMAM: low level breakdown



# Supercomputers

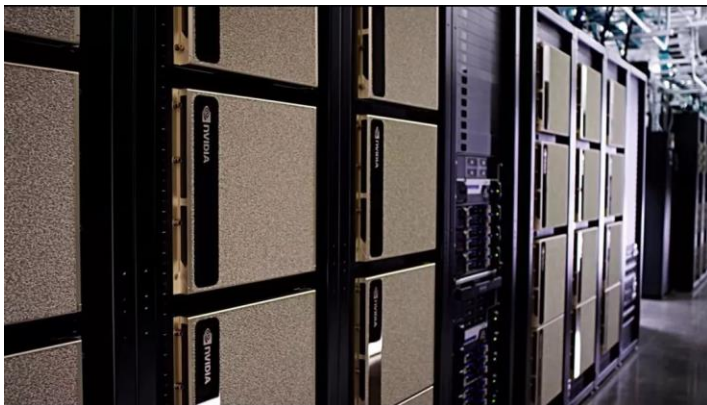
**#9\* Marconi100** – IBM Power9 + Nvidia V100



**#47\* Marconi** – Intel Xeon 8160 SkylakeX



**#7\* Selene** – AMD 7742 Epyc + Nvidia A100 (DGX)



**#203\* Astra** – ARM Cavium ThunderX2 (ARMIDA@E4)



*\*at ranking time*

# Why did we select these supercomputers?

CPU	ISA	Cores	SMT	Operating Frequency (GHz)	Peak IPC	Peak FLOPs (GFLOPs)	Peak Memory Bandwidth (GB/s)	TDP (W)	Peak Energy Efficiency (GFLOPs/W)	Characteristic
Intel Xeon 8160	x86-64	24	OFF	2.1 (AVX-512)	4 (6)	1612	120	150	10.75	Unbalance on the compute
IBM Power 9	Power ISA	16	x4	3.8	6	486	160	250	1.94	Unbalance on the memory
Cavium ThunderX2	ARMv8.1	32	OFF	2.5	4	640	160	200	3.20	Unbalance on the memory
AMD 7742 Epyc	x86-64	64	OFF	2.25 (AVX-256)	4 (8)	2662	190	225	11.82	Good overall balance

\*nodes of all the systems are double sockets



# Applications & Software Stack

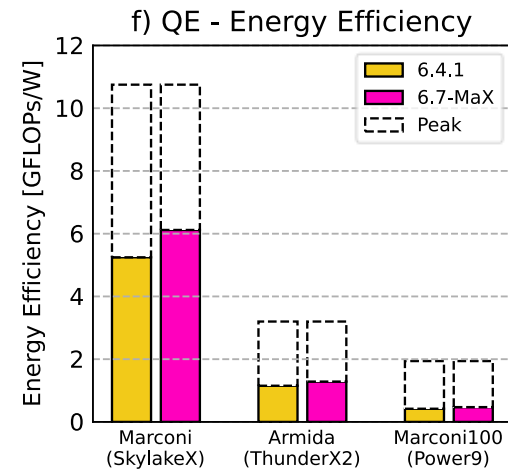
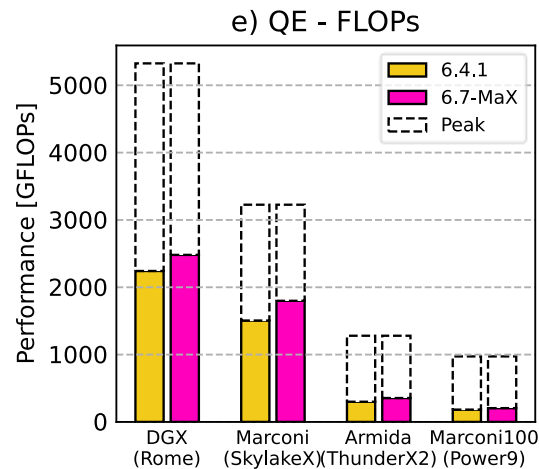
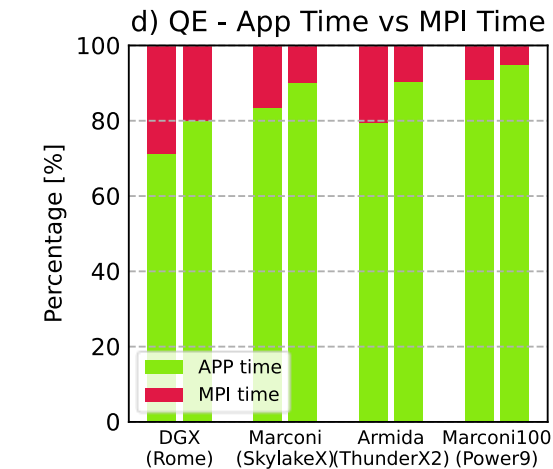
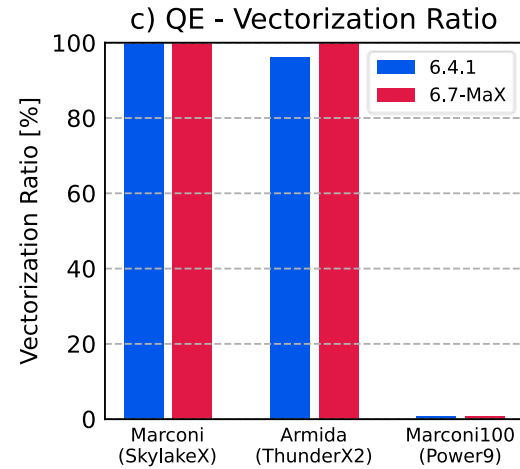
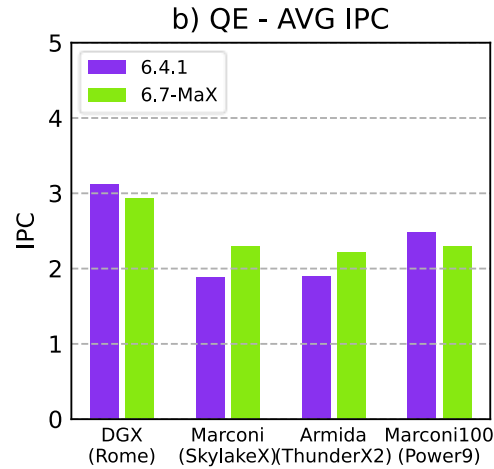
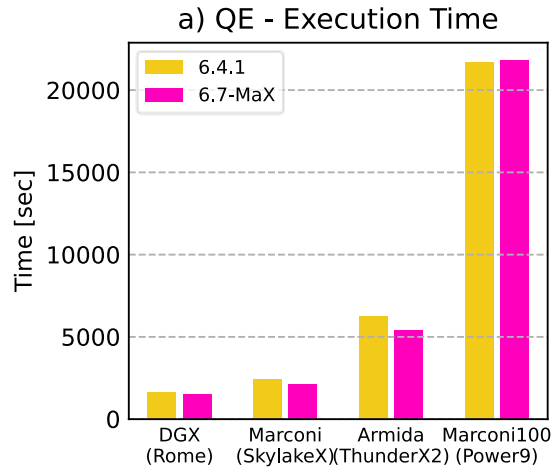


CPU	Compiler	MPI	BLAS & LAPACK	ScaLAPACK	FFTW
Intel Xeon 8160	GCC 9.3	OpenMPI 4.1.1	MKL 2018	MKL 2018	MKL 2018
IBM Power 9	GCC 9.3	OpenMPI 4.1.1	ESSL 6.2.1 OpenBLAS 0.3.18	Netlib-scalapack 2.1.0	FFTW 3.9
Cavium ThunderX2	GCC 9.3	OpenMPI 4.1.1	ArmPL 20.3	Netlib-ScaLaPACK2.1.0	FFTW 3.9
AMD 7742 Epyc	GCC 9.3	OpenMPI 4.1.1	AMDblis/Flame 3.0	AMDScaLaPACK 3.0	AMDFFTW 3.0

**Other accessory libraries:**

- Netcdf Fortran 4.5.3
- Netcdf C 4.8.1
- HDF5 1.10.7
- ELSI 2.7.1





### Configuration:

- Single node experiments
- Pure MPI (one MPI rank per core)
- We compared two different version of the code

### Behaviors:

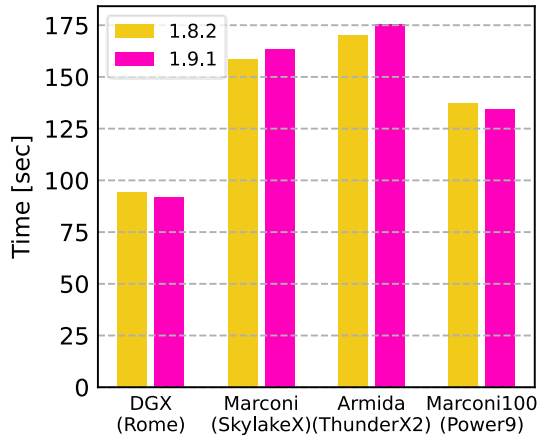
- *Execution time*: proportional with the peak FLOPS (Power9?)
- *IPC*:  $\geq 2$  limited by the number of vector ALUs (2)
- *Vectorization ratio*:  $\geq 100\%$  (Power9?)
- FLOPS and FLOPS/W: proportional with the peak FLOPS

### Conclusions:

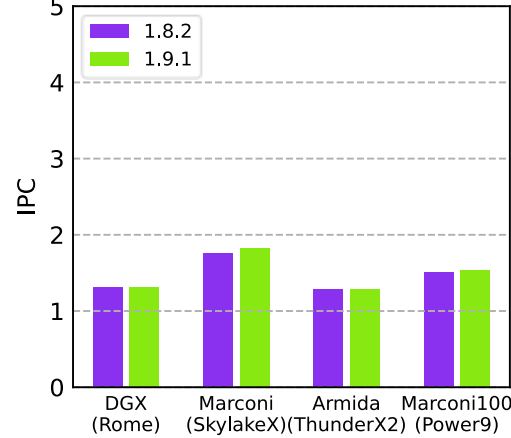
- QE is compute bound!
- High vectorized codes work pretty well!
- Intel parchs ❤️ QE



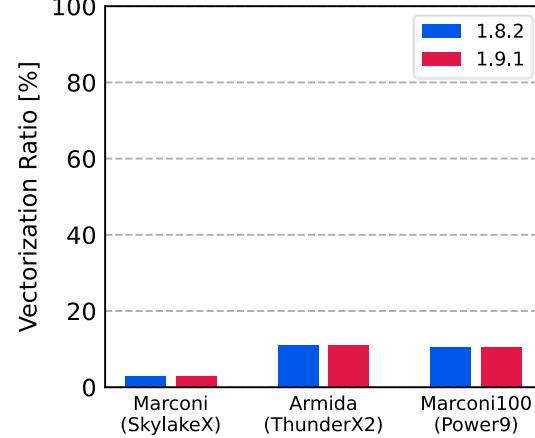
a) BigDFT - Execution Time



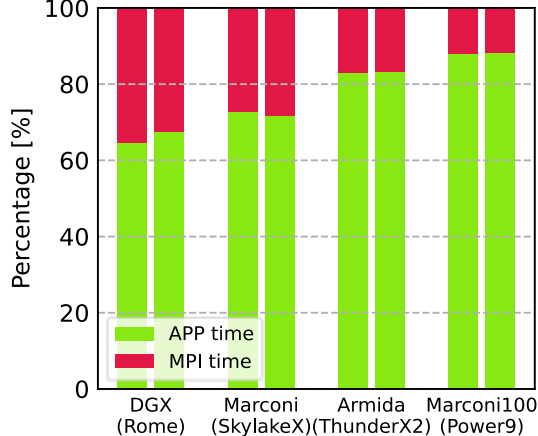
b) BigDFT - AVG IPC



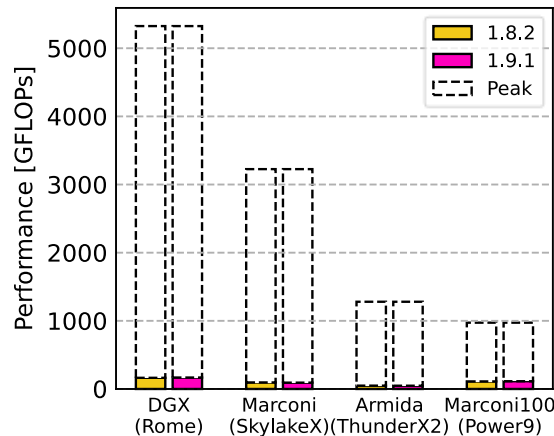
c) BigDFT - Vectorization Ratio



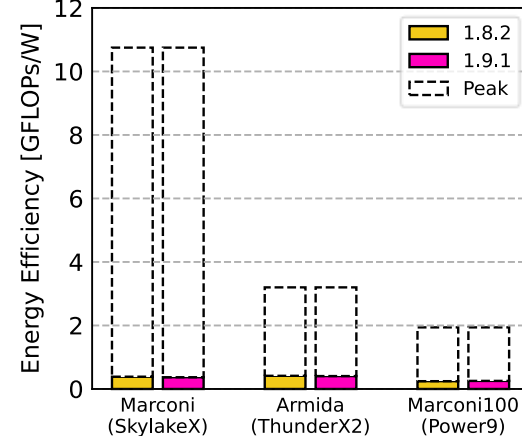
d) BigDFT - App Time vs MPI Time



e) BigDFT - FLOPs



f) BigDFT - Energy Efficiency



## Configuration:

- Single node experiments
- MPI & Openmp
- We compared two different version of the codes

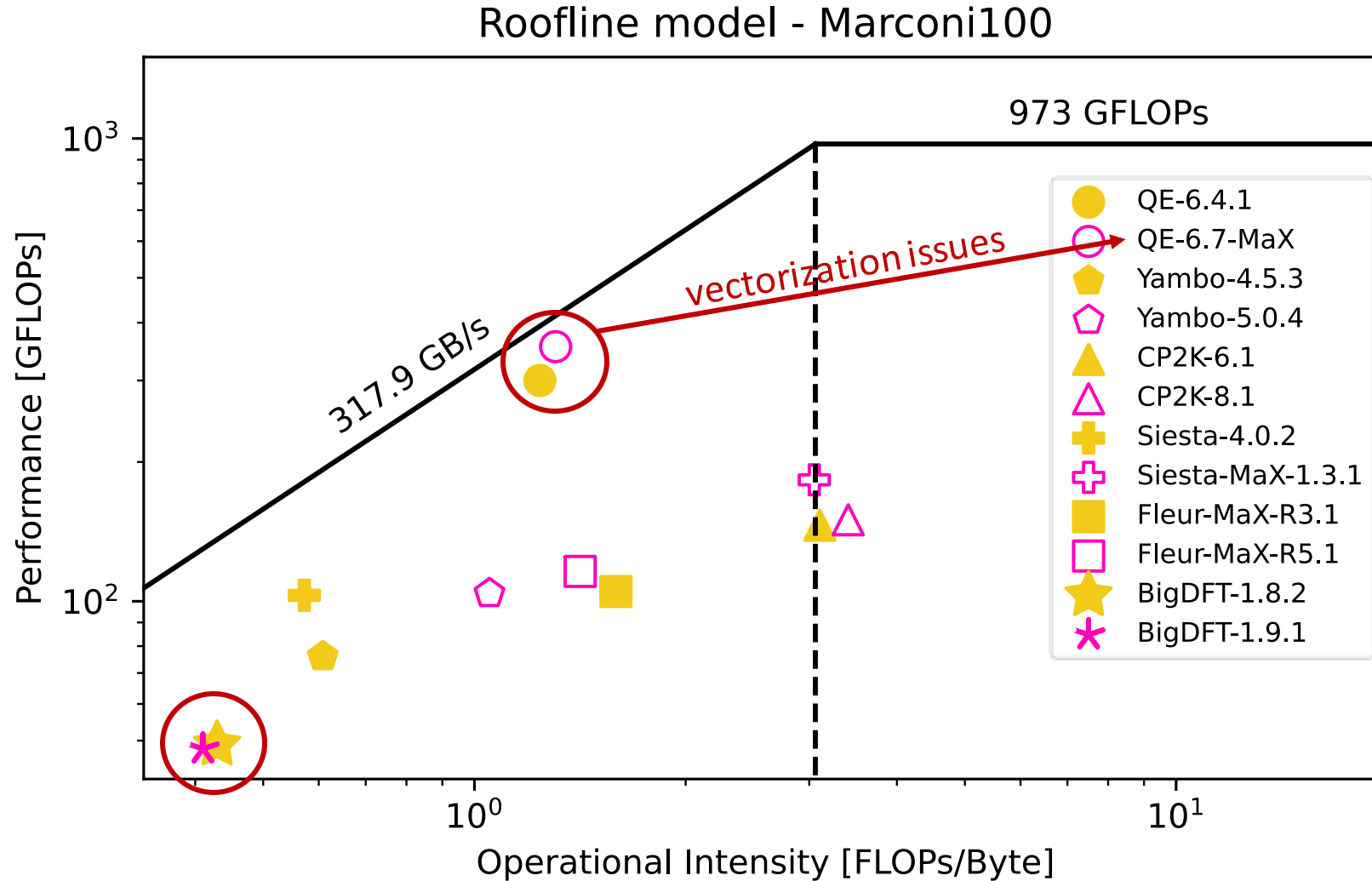
## Behaviors:

- *Execution time*: proportional with the peak memory bandwidth and the capacity to move data!
- *IPC*: <2 limited by the memory
- *Vectorization ratio*: <15% this workload is very difficult to vectorize!
- FLOPS and FLOPS/W: limited by the memory, are these good metrics?

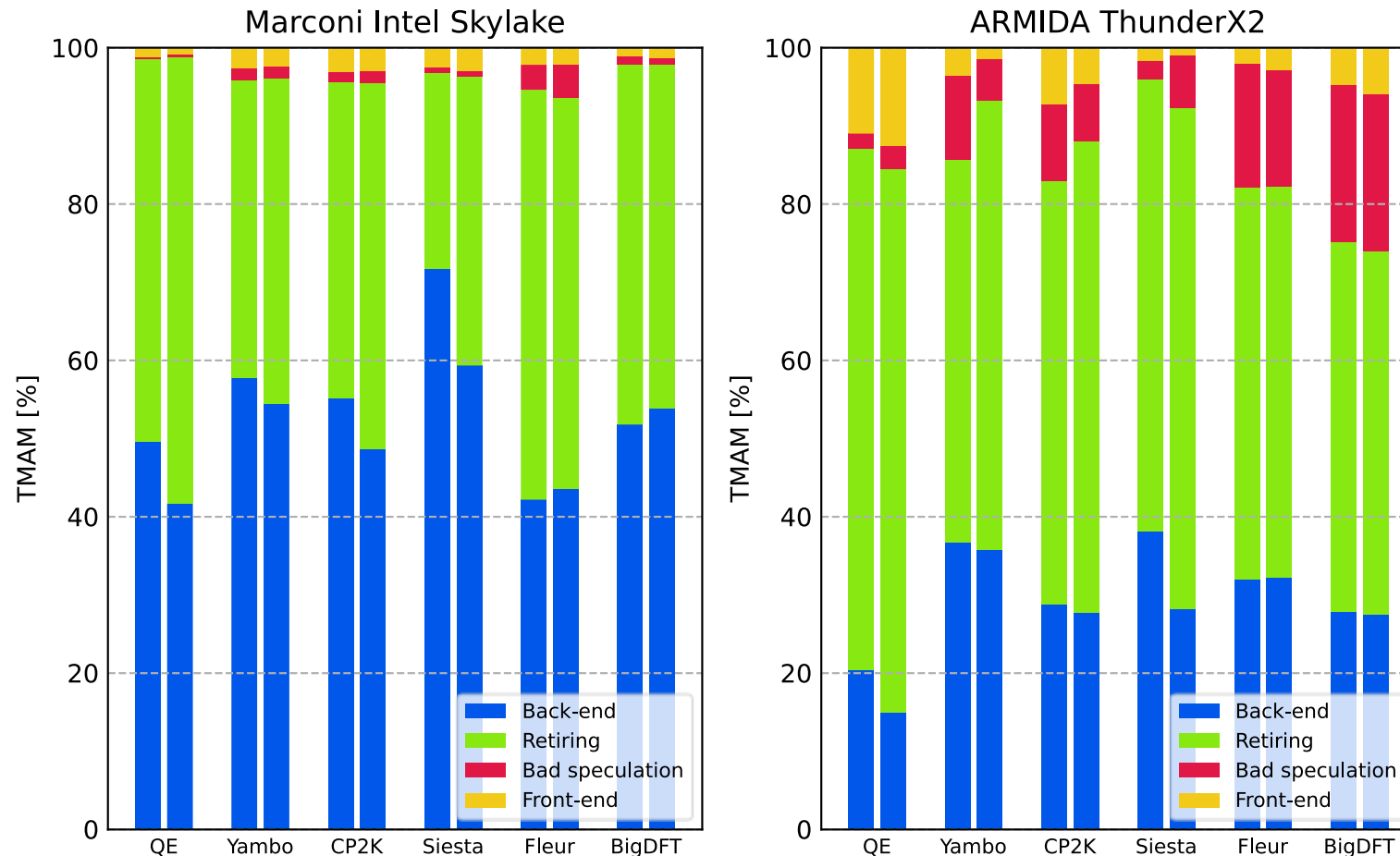
## Conclusions:

- BigDFT is memory bound!
- IBM Power9 ❤️ BigDFT

# Roofline – IBM Power9



# TMAM – Intel SkylakeX vs ARM ThunderX2



\*single node experiment results

## Conclusions & Take away messages

- *Application workload have different performance on different architectures*
- *Lack of microarchitecture efficiency limits the application performance more than scalability*
- *Performance models can help you to understand how application use the system resources*
- *Tools may help to spot inefficiencies, but you do the thinking!*

William H. McRaven: “If you can't do the little things right, **you will never do the big things right.**”

But what about GPUs?

