



**Barcelona
Supercomputing
Center**

Centro Nacional de Supercomputación



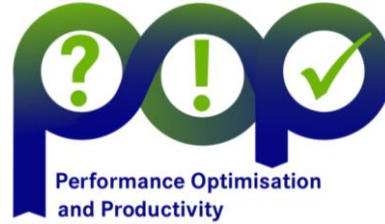
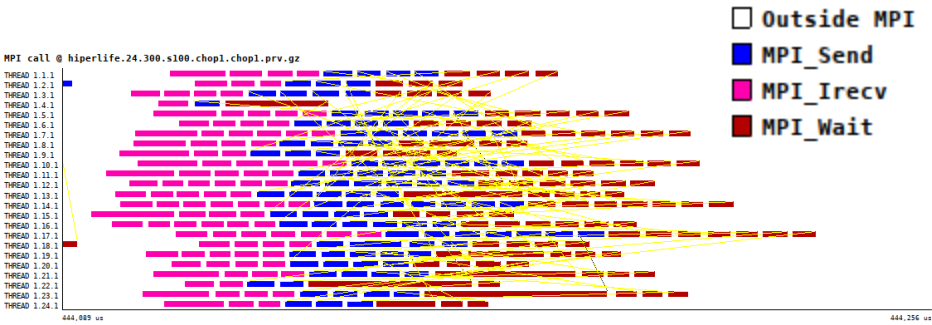
**EXCELENCIA
SEVERO
OCHOA**

Evolving HPC Architectures: From CPU-Centric to Post-GPU Systems

14 April 2026

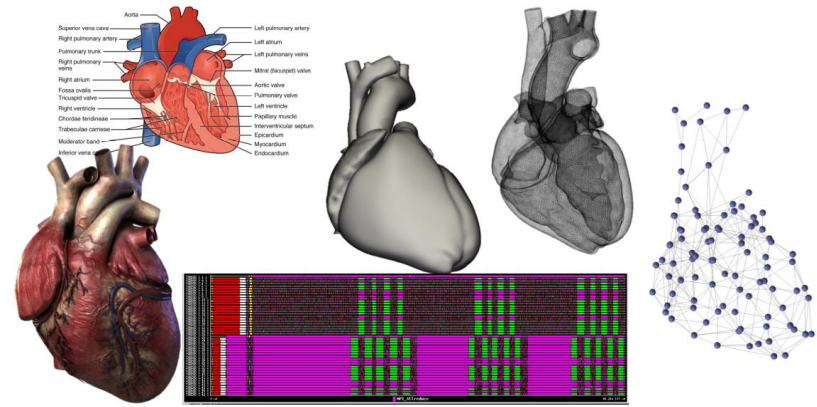
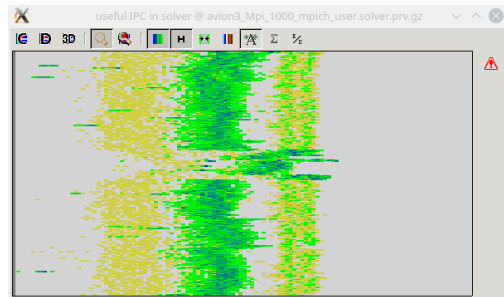
Marta Garcia Gasulla

My biases

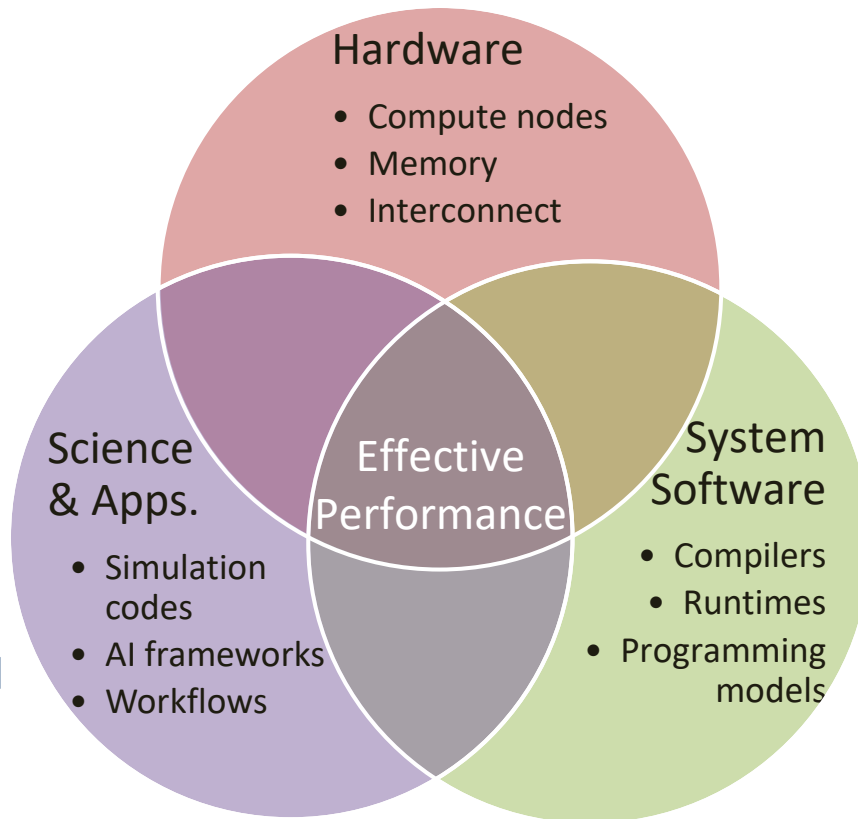



10 years and +300 HPC codes analyzed


+20 years in HPC

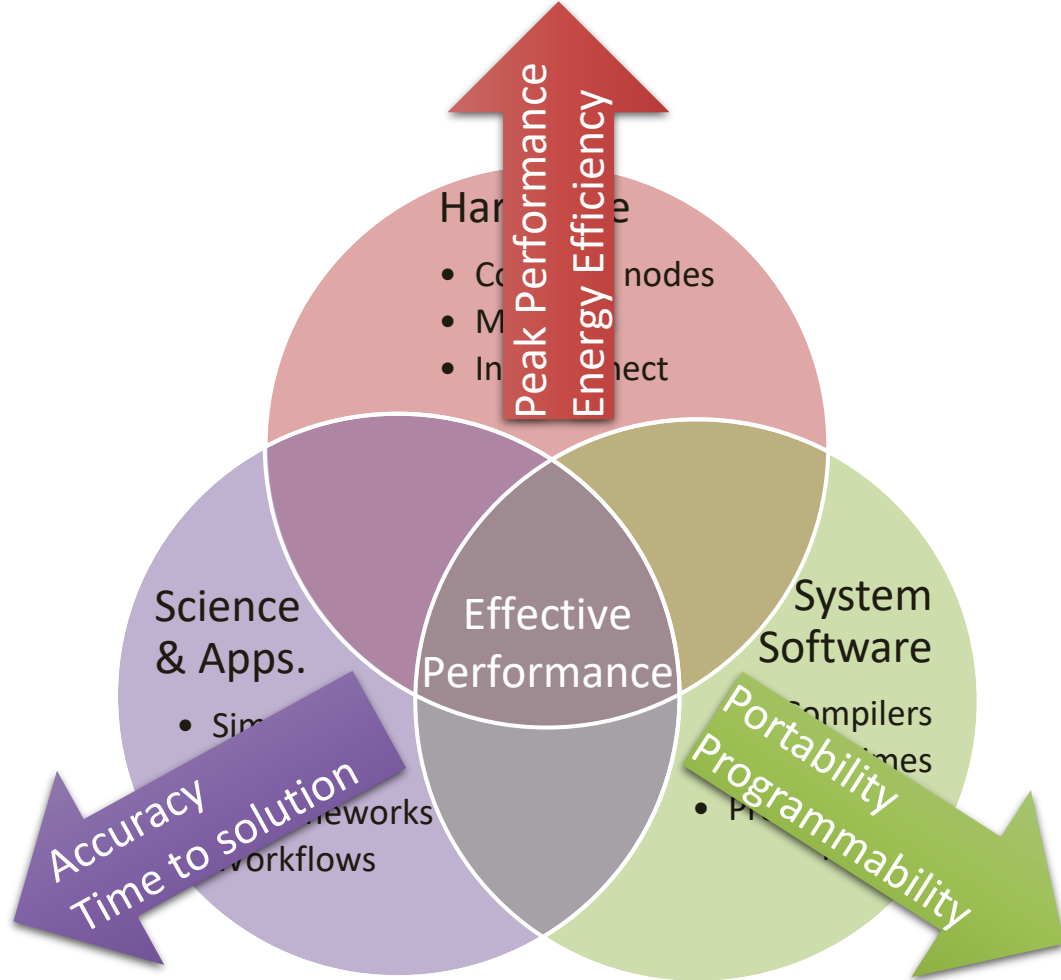


⚡ Defines
what is possible



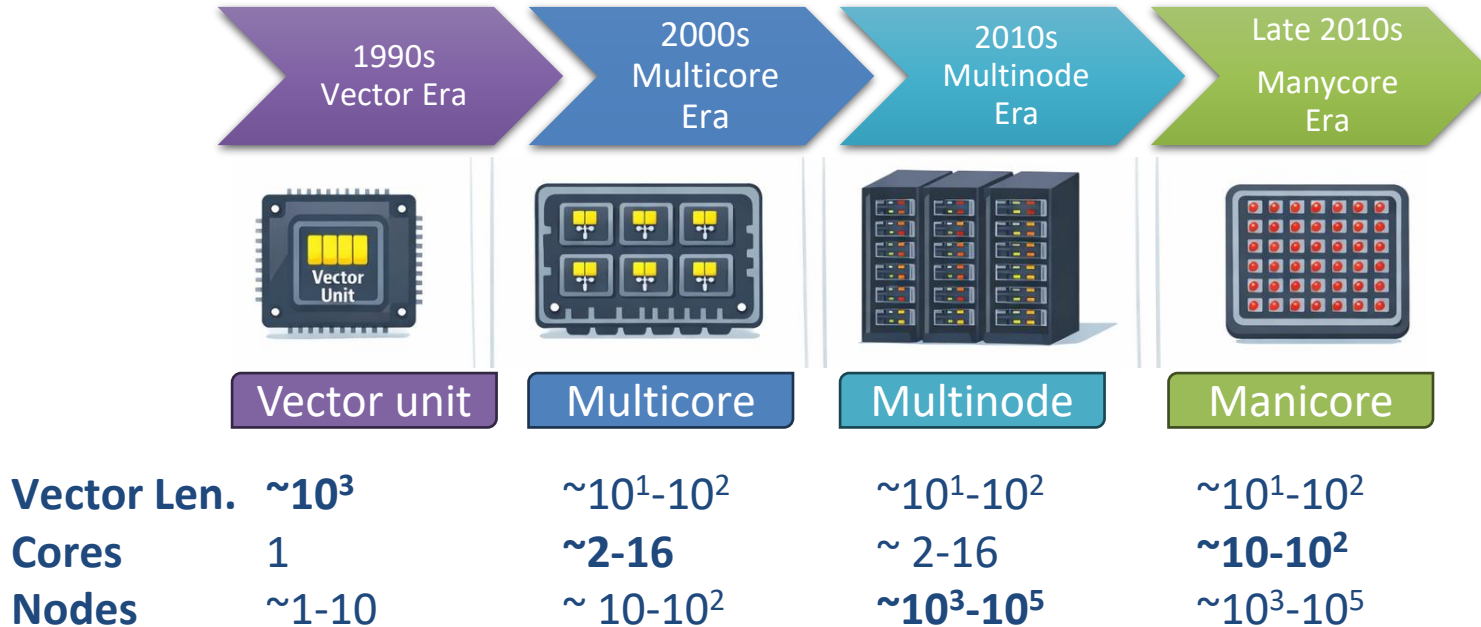
 Defines
what is needed

 Defines
what is usable



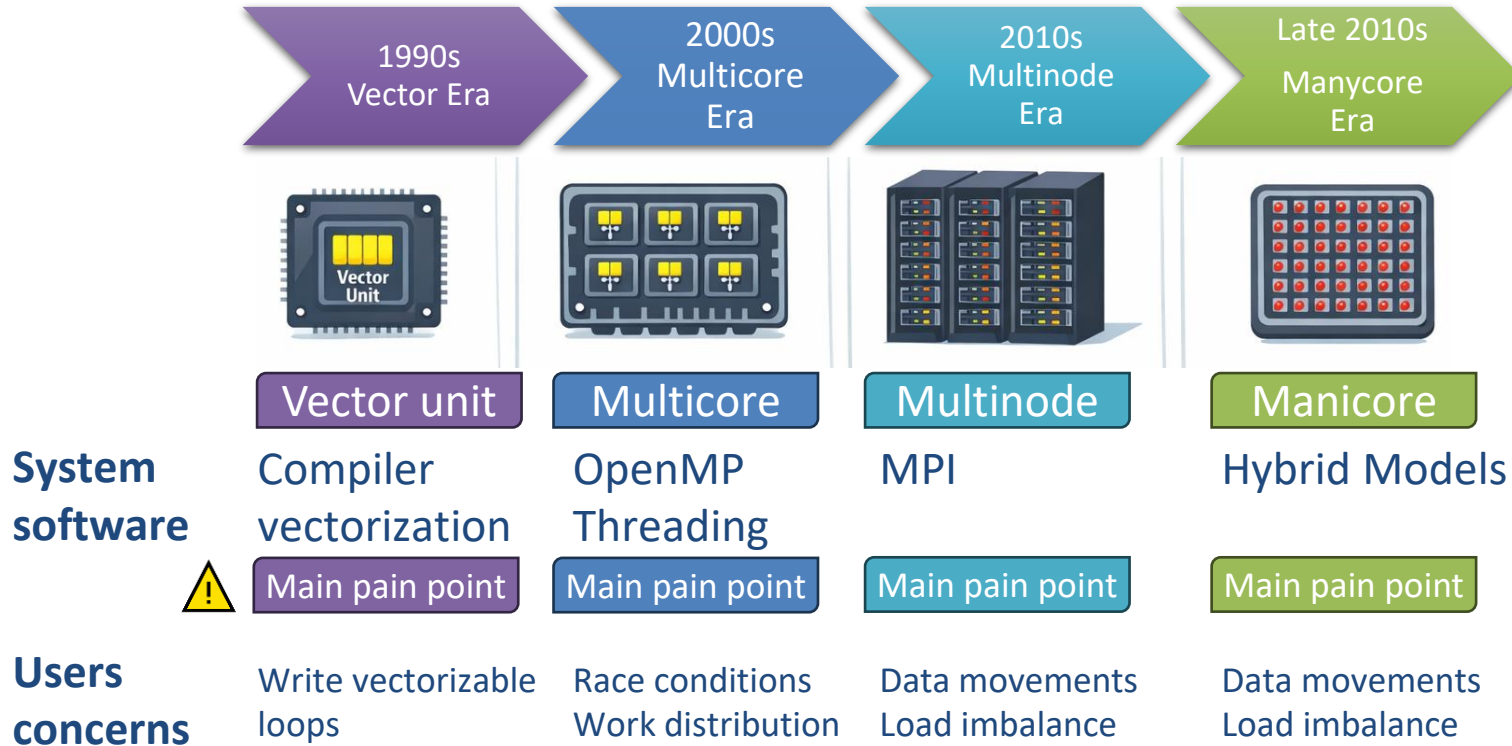
Performance is achieved when all three are aligned.

Where do the FLOPs come from in the CPU era?



The Era in which performance was achieved by putting more

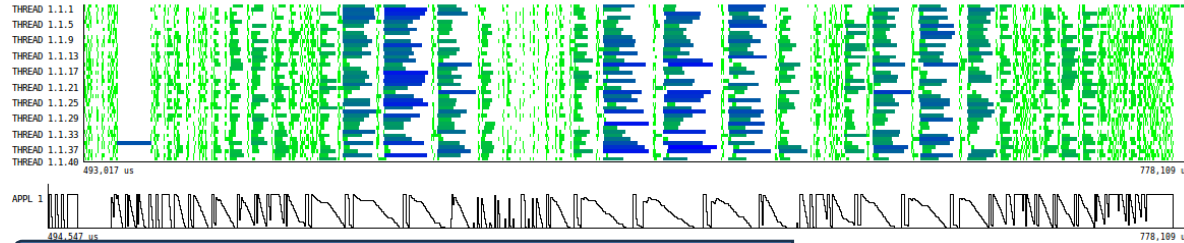
As hardware scaled, software had to follow



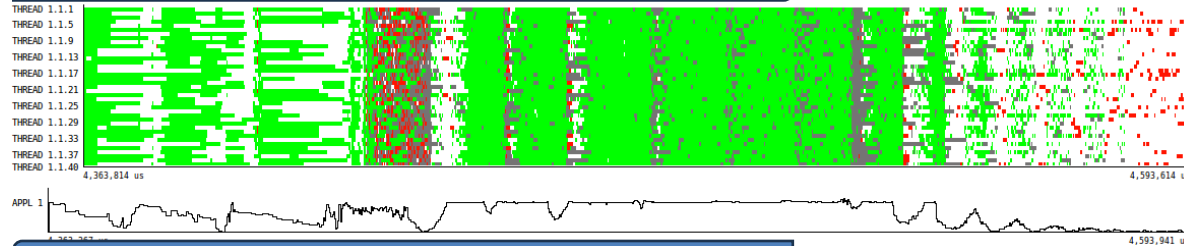
System software acted as a “shock absorber”

A multicore-manycore example

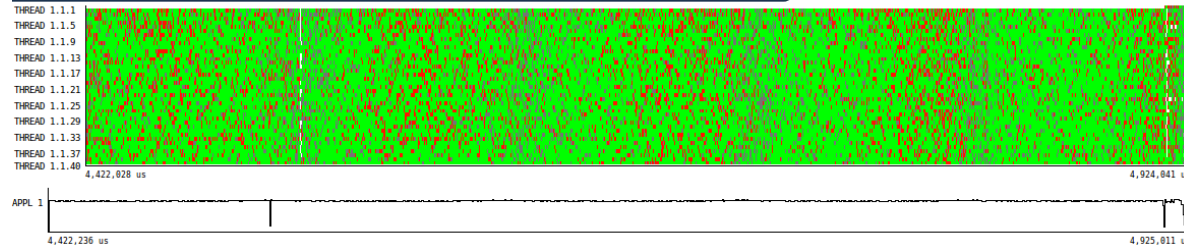
Original code based on OpenMP parallel loops



First taskification of the code



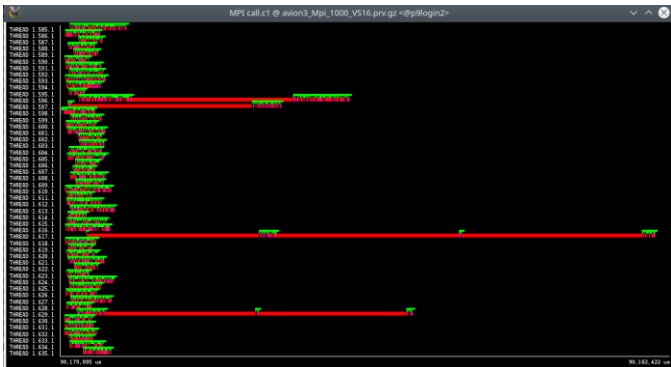
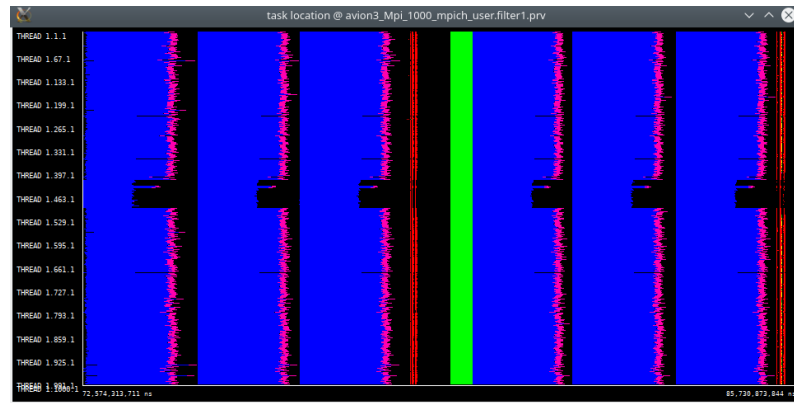
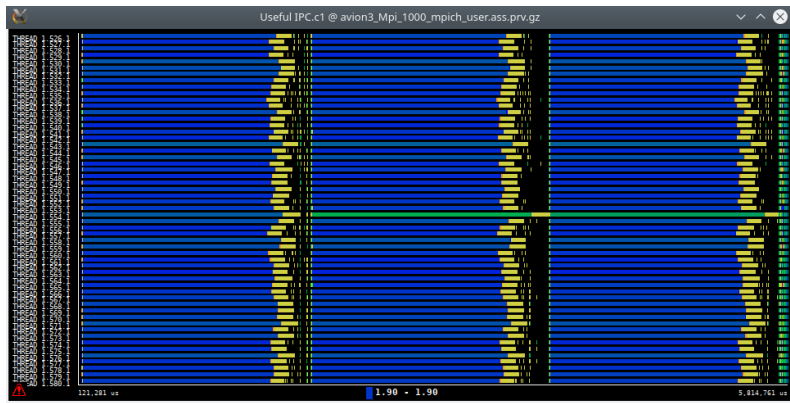
Optimized taskified code: Priorities + nested tasks



OpenMP includes: **User:**

- Tasking model
 - Nested tasks
 - Priorities
- Taskifies code
 - Adds nested parallelism
 - Adds tasks priorities

Multinode → multiimbalance



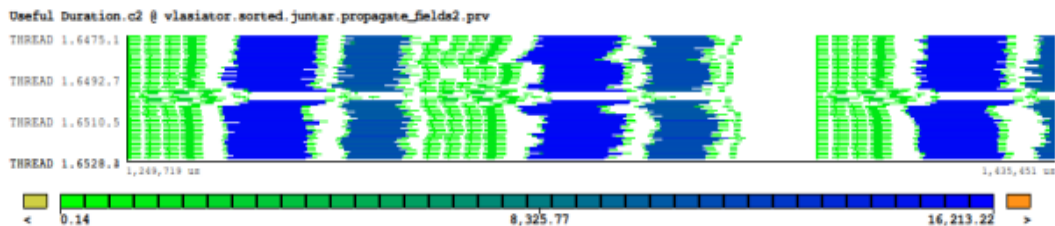
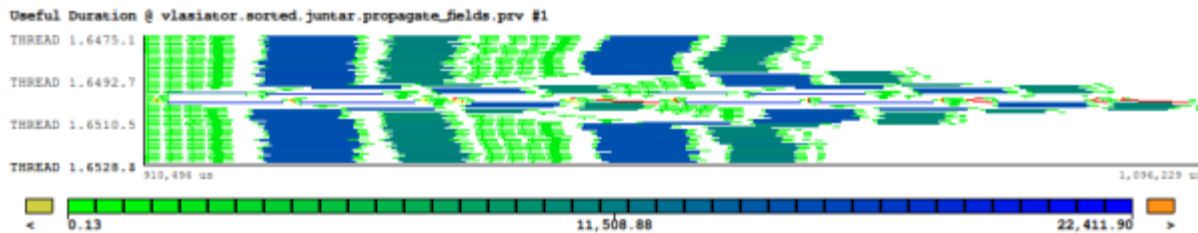
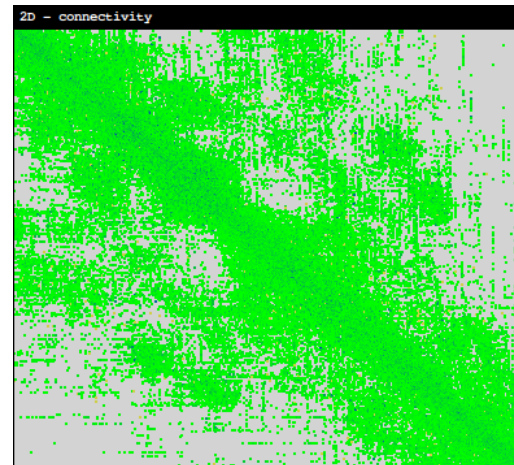
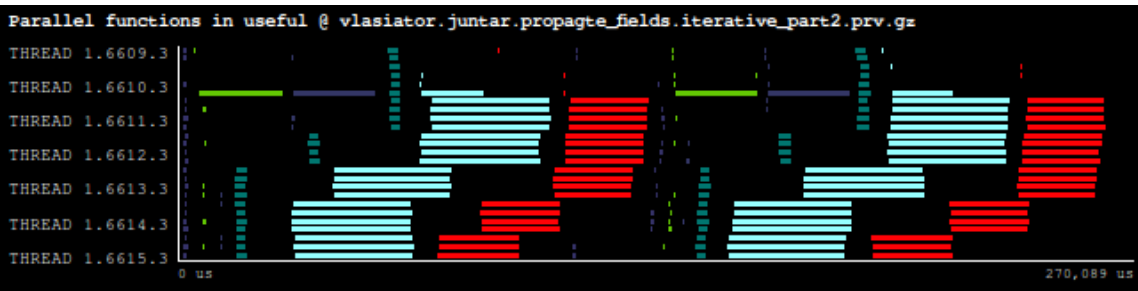
Single execution
→ 3 different regions
→ 3 different
imbalance issues



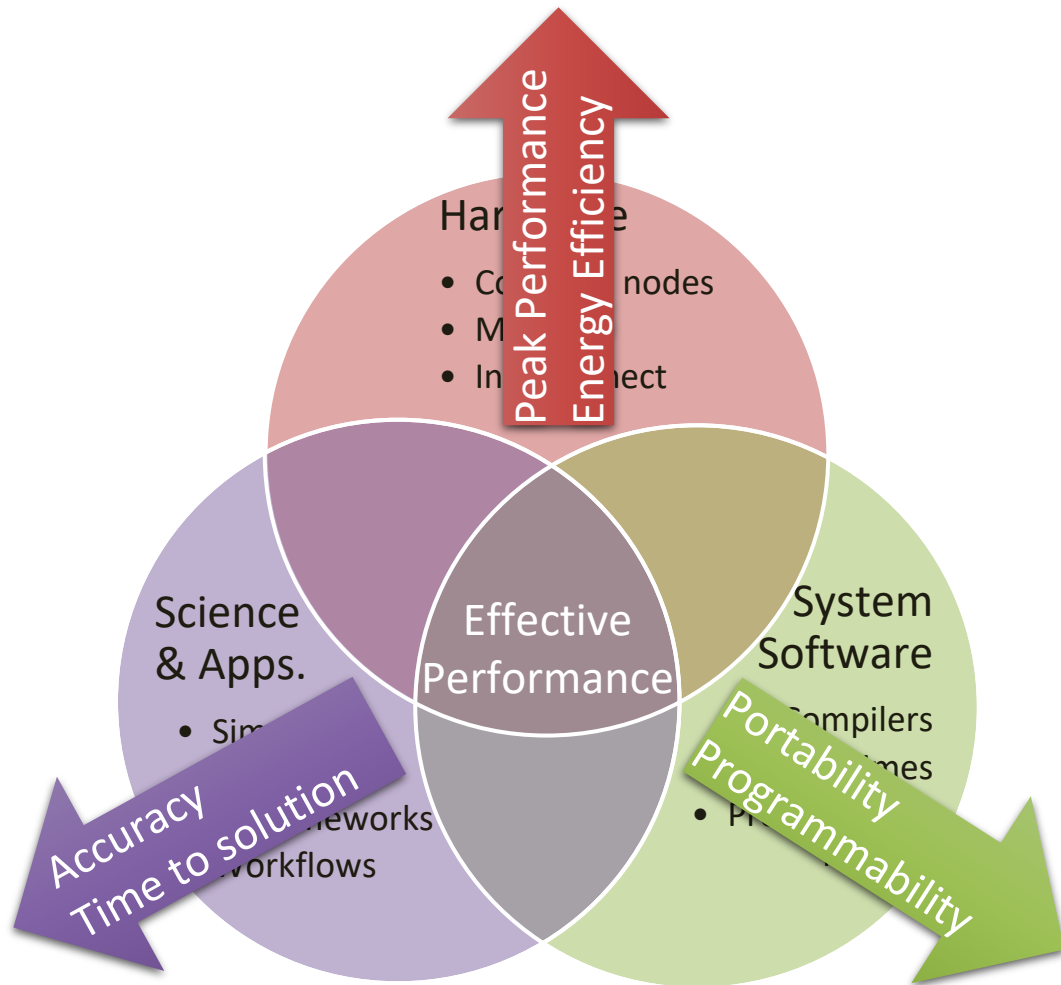
Go hybrid!

Load imbalance at different levels

>1PB of data being moved



More complex to orchestrate.
But different levels can
alleviate each other pain point



Software had to follow...

... by sacrificing portability, programmability, and accuracy

As hardware scaled, software had to follow



- General purpose cores
- Incremental evolution
- User in charge of expressing parallelism
- System software abstracts HW details

- Designed for FLOPs, not programmability
- Hardware-driven evolution

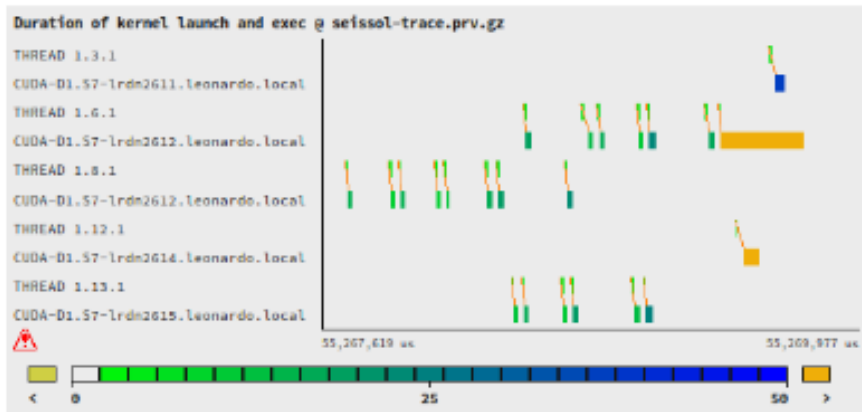
- Thousands of light-weight specialized computing units
- Hardware details exposed to the programmer
- No standards or system software hardware agnostic ready

Until now, software adapted gradually.
With GPUs... adaptation became a rewrite.

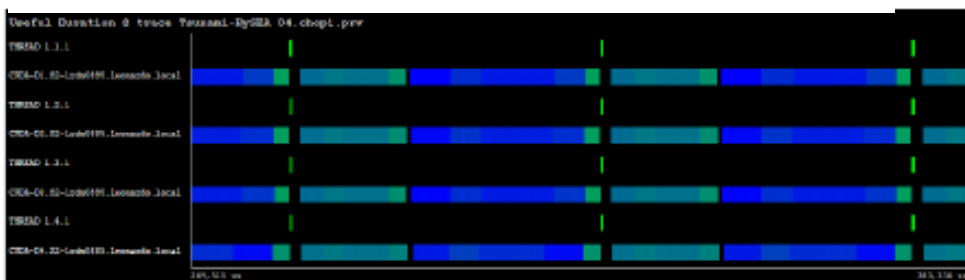
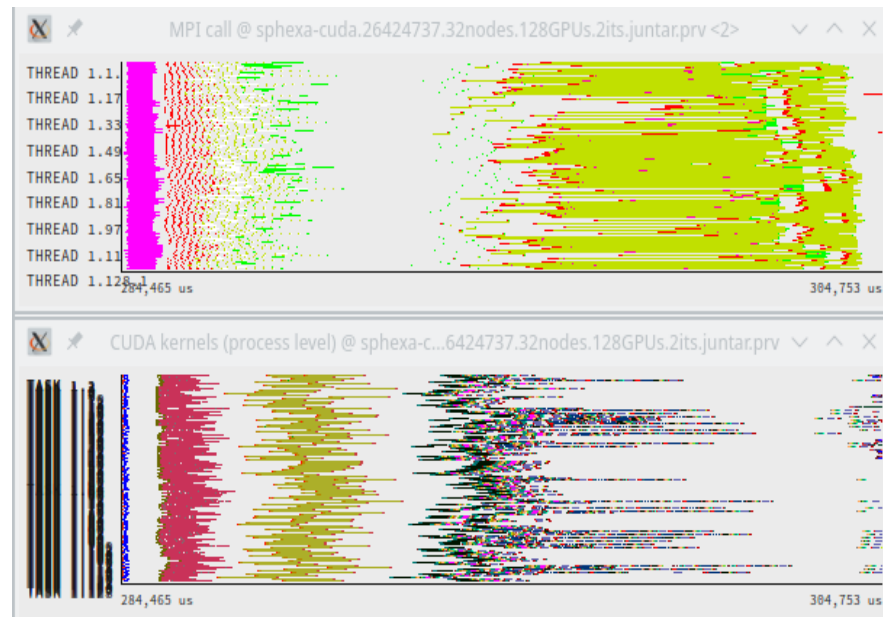
How GPU codes looked like

How GPU codes look like

GPU kernels too fine grained



Struggling to orchestrate all the resources



All work offloaded to GPUs,
CPUs sit idle...



in AI-land...

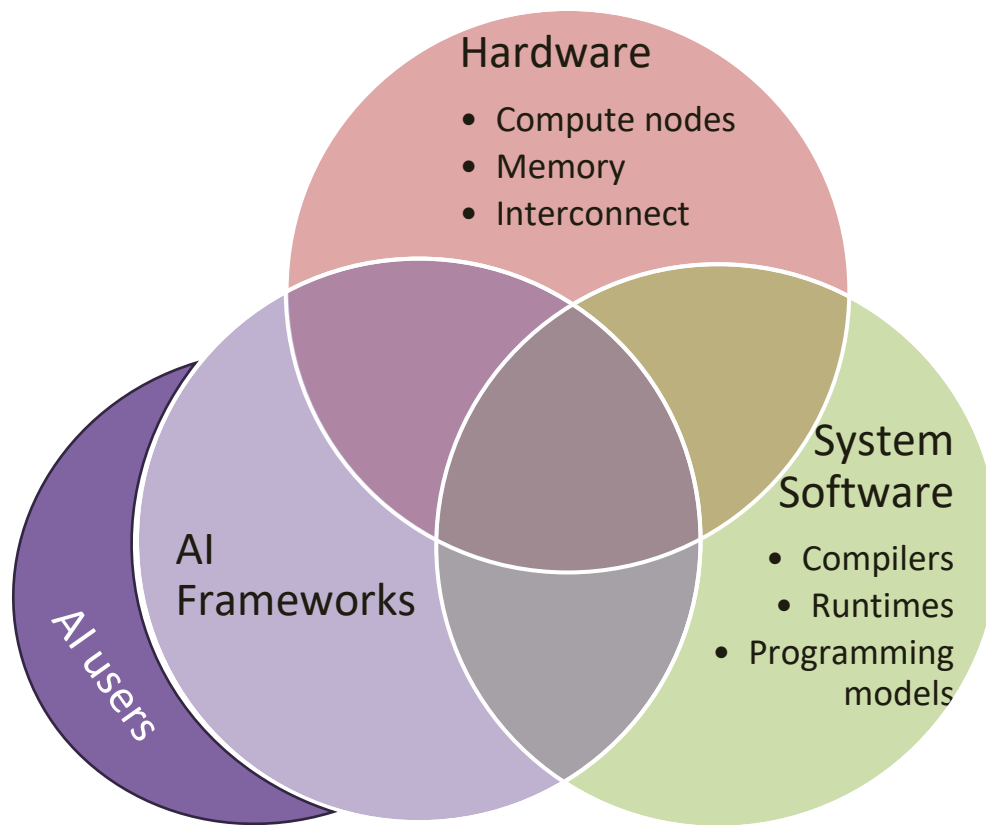
AI Needed GPUs to Bloom



AI Evolution



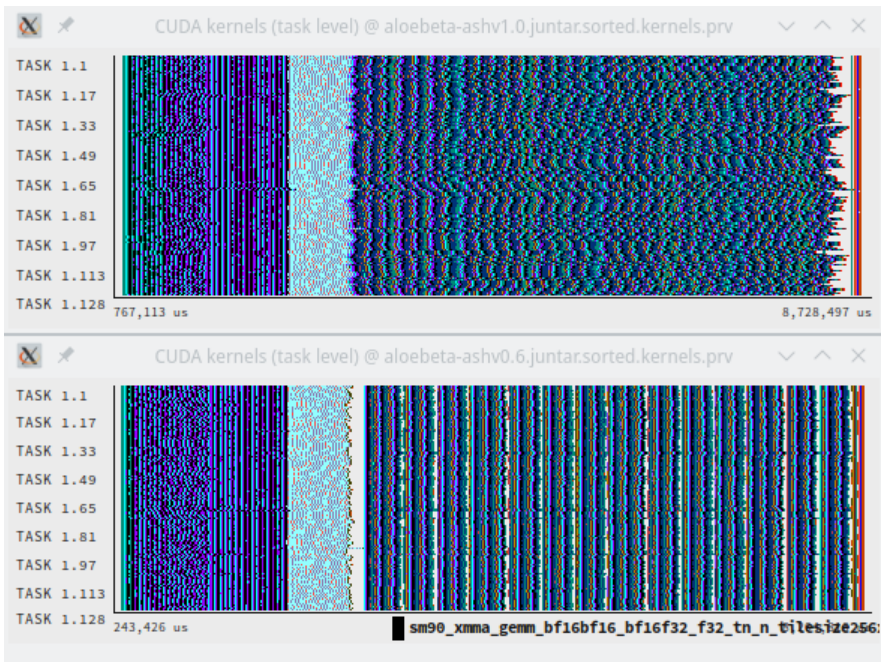
When GPUs arrived, the same ideas suddenly became practical.



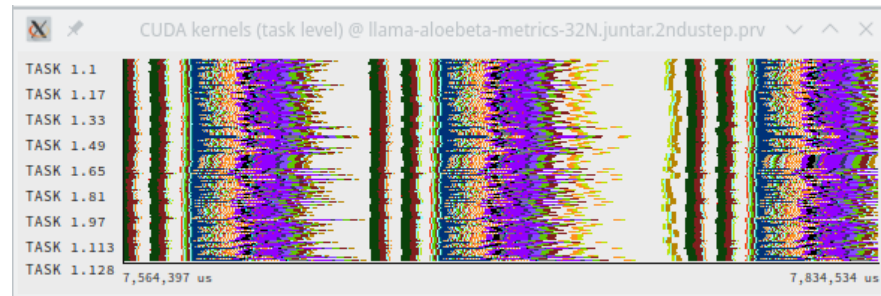
AI Frameworks made it possible isolating the user from hardware and software

What happens inside an AI framework?

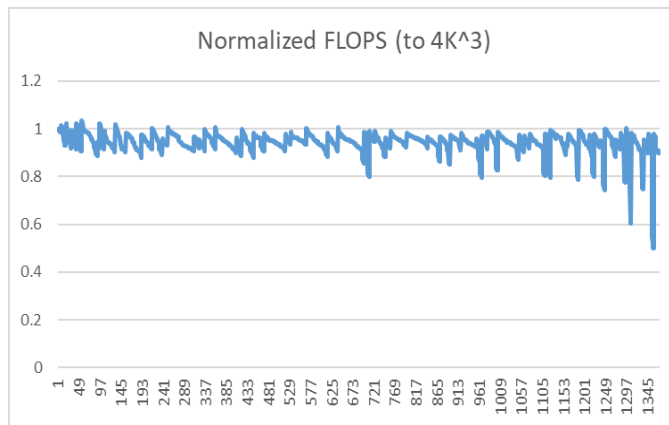
To overlap or not to overlap...
that is the question

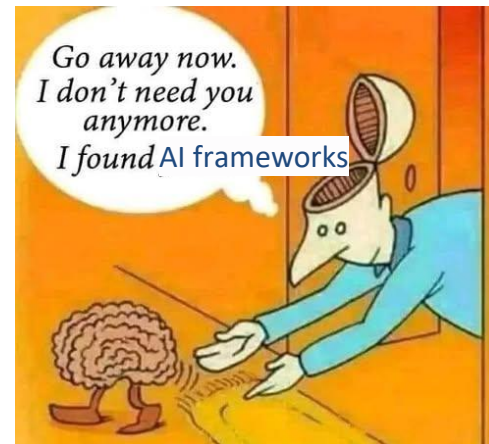
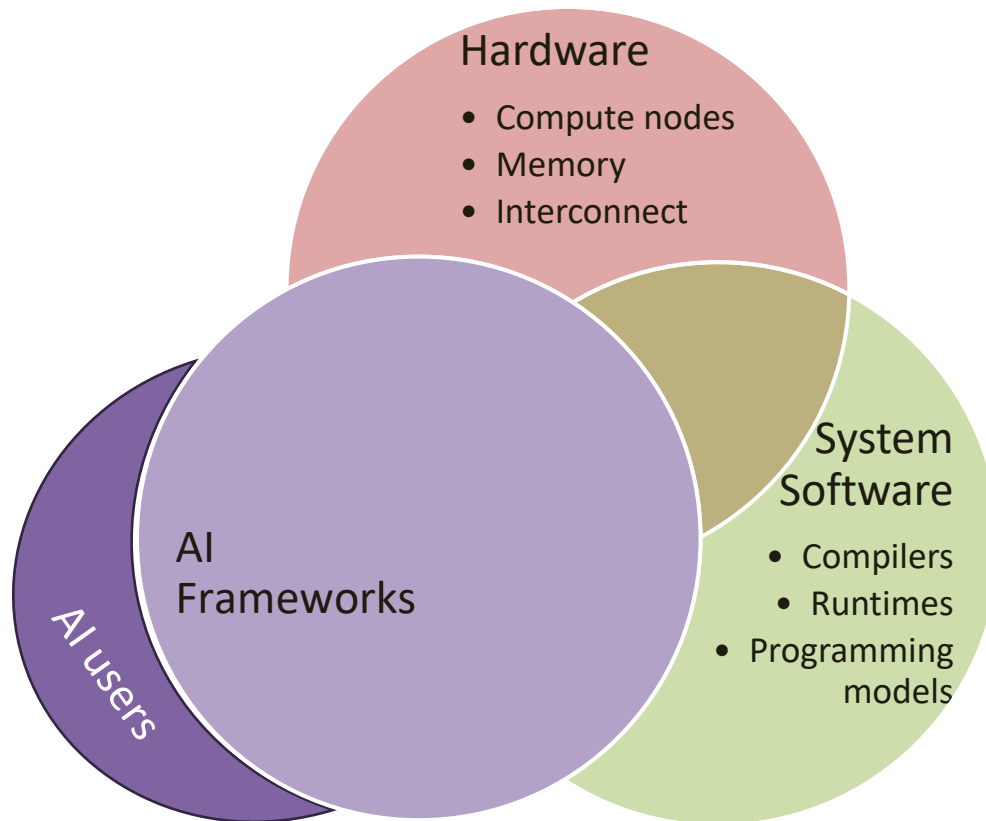


Imbalance!



Different performance for different GEMM sizes, but who controls them?





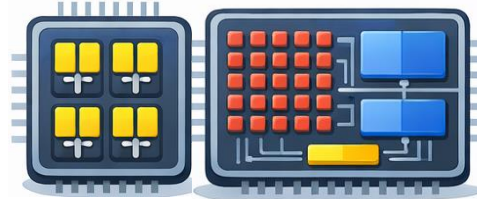
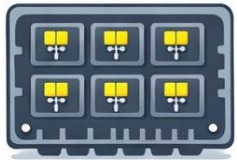
AI Frameworks are blocking user from the performance and efficiency too

Where do we go from here?

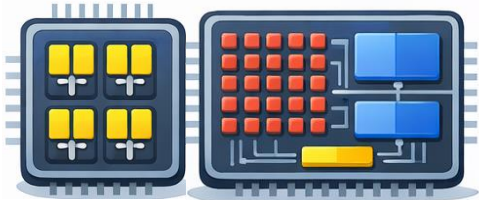
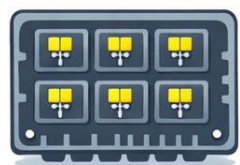
CPUs Era

Change
paradigm

GPUs Era



Converging



EPAC: European Initiative Accelerator

- A core that can work independent
- A very long vector unit.

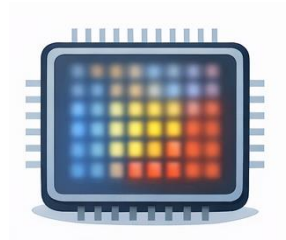
GPUs evolve towards more general-purpose execution

Latest GPU systems

- CPU and GPUs tightly integrated

Converging

The boundary between CPU and GPU is not disappearing it is becoming an implementation detail.



Will system software be able to hide this implementation detail?

Wrap up

- We need the three actors to be aligned to achieve effective performance: Hardware, System Software and Applications
- De advent of GPUs meant a major disruption
 - Rewrite and lose of portability and abstraction for HPC codes
 - Bloom for AI
- Where are we going now?
 - Converging to more integrated CPUs and GPUs
 - Will the system software be able to keep up and abstract the user from hardware details?



**Barcelona
Supercomputing
Center**

Centro Nacional de Supercomputación



**EXCELENCIA
SEVERO
OCHOA**

Thank you

marta.garcia@bsc.es

Best Practices for Performance and
Programmability