



The Abdus Salam  
International Centre  
for Theoretical Physics



IAEA  
International Atomic Energy Agency

# Foundation of Parallel Systems for High-Performance Computing

**Ivan Girotto – [igirotto@ictp.it](mailto:igirotto@ictp.it)**

Information & Communication Technology Section (ICTS)  
International Centre for Theoretical Physics (ICTP)



# Why use Computers in Science?

- Use complex theories without a closed solution: solve equations or problems that can only be solved numerically, i.e. by inserting numbers into expressions and analyzing the results
- Do “impossible” experiments: study (virtual) experiments, where the boundary conditions are inaccessible or not controllable
- Benchmark correctness of models and theories: the better a model/theory reproduces known experimental results, the better its predictions



# What is High-Performance Computing (HPC)?

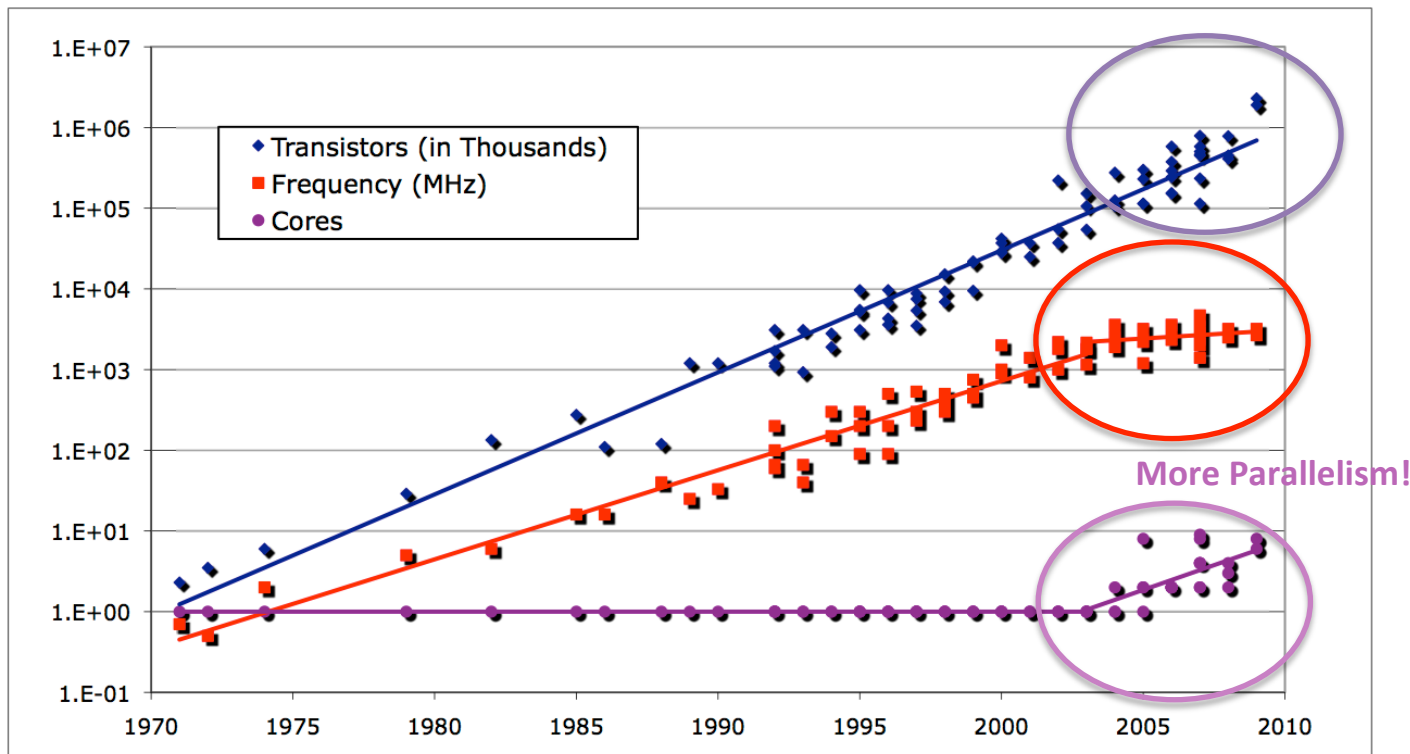
- Not a real definition, depends from the prospective:
  - HPC is when I care how fast I get an answer
- Thus HPC can happen on:
  - A workstation, desktop, laptop, smartphone!
  - A supercomputer
  - A Linux Cluster
  - A grid or a cloud
  - Cyberinfrastructure = any combination of the above
- HPC means also **High-Productivity Computing**



# Why would HPC matter to you?


- Scientific computing is becoming more important in many research disciplines
- Problems become more complex, thus need complex software and teams of researchers with diverse expertise working together
- HPC hardware is more complex, application performance depends on many factors
- Technology is also for increasing competitiveness

# CPUs Trend and Moore's Law



Strongly market driven  Mobile, Tv set, Screens  
Video/Image processing

Intel  New arch to compete with ARM  
Less Xeon, but PHI

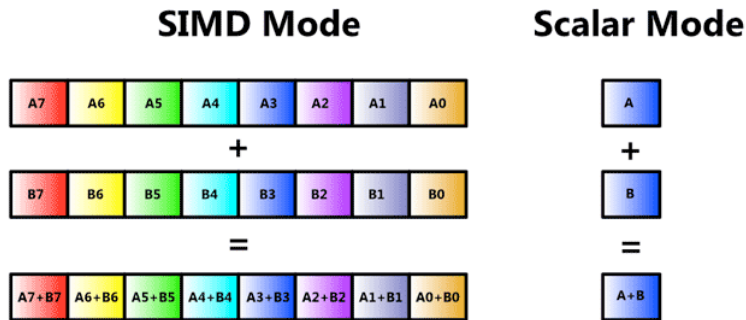
ARM  Main focus on low power mobile chip  
Qualcomm, Texas inst. , Nvidia, ST, ecc  
new HPC market, server market

NVIDIA  GPU alone will not last long  
ARM+GPU, Power+GPU

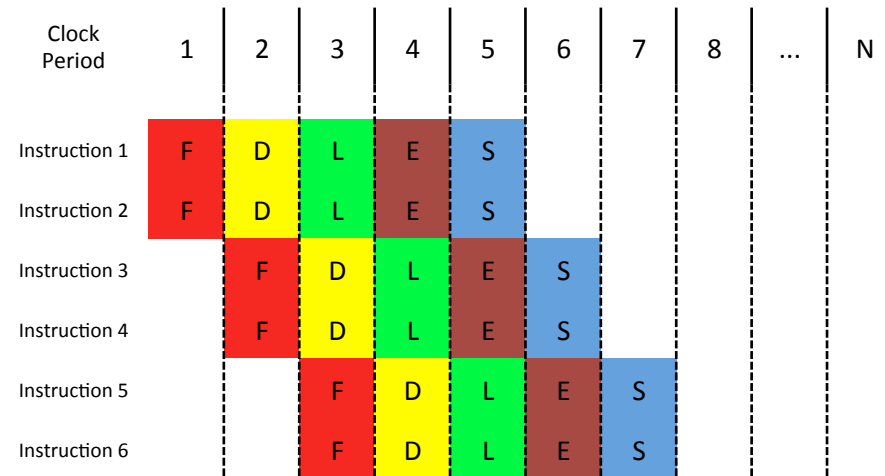
IBM  Embedded market  
Power+GPU, only chance for HPC

AMD  Console market  
Still some chance for HPC

# To the Extreme - Parallel Inside



Vector Units for processing multiple data in //



Pipelined/Superscalar design: multiple functional units operate concurrently

# Consequences

- Parallelism is no longer only an option for either thinking bigger or improve the time to solution
- It is inescapable to not disadvantage of the next generation of processors and compute systems





# The CPU Memory Hierarchy



The diagram illustrates the CPU memory hierarchy as a pyramid with three levels. The top level is a teal triangle labeled 'CPU Registers'. The middle level is a red trapezoid labeled 'CACHE'. The bottom level is a dark blue trapezoid labeled 'MAIN MEMORY'. To the right of the pyramid, there is a teal horizontal bar labeled 'COMPUTATION' at the top and a yellow horizontal bar labeled 'APPLICATION DATA' at the bottom. A large, red, double-headed arrow with a diagonal hatching pattern connects the 'COMPUTATION' bar to the 'APPLICATION DATA' bar, indicating the flow of data between these two stages.

CPU  
Registers

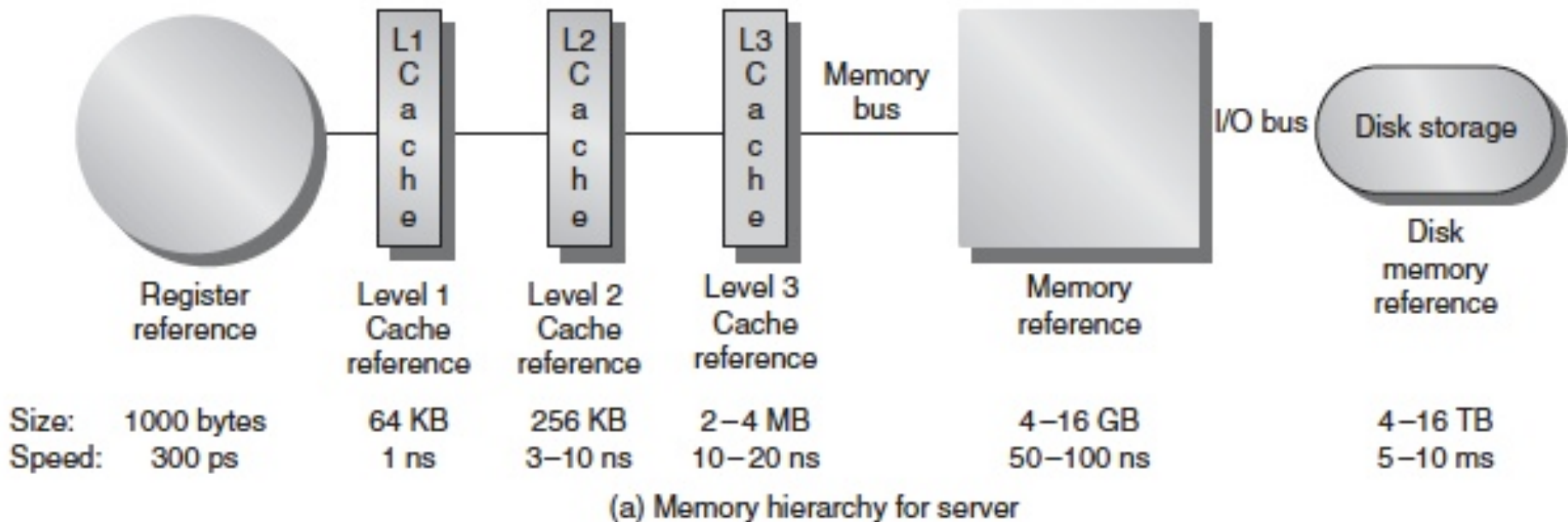
CACHE

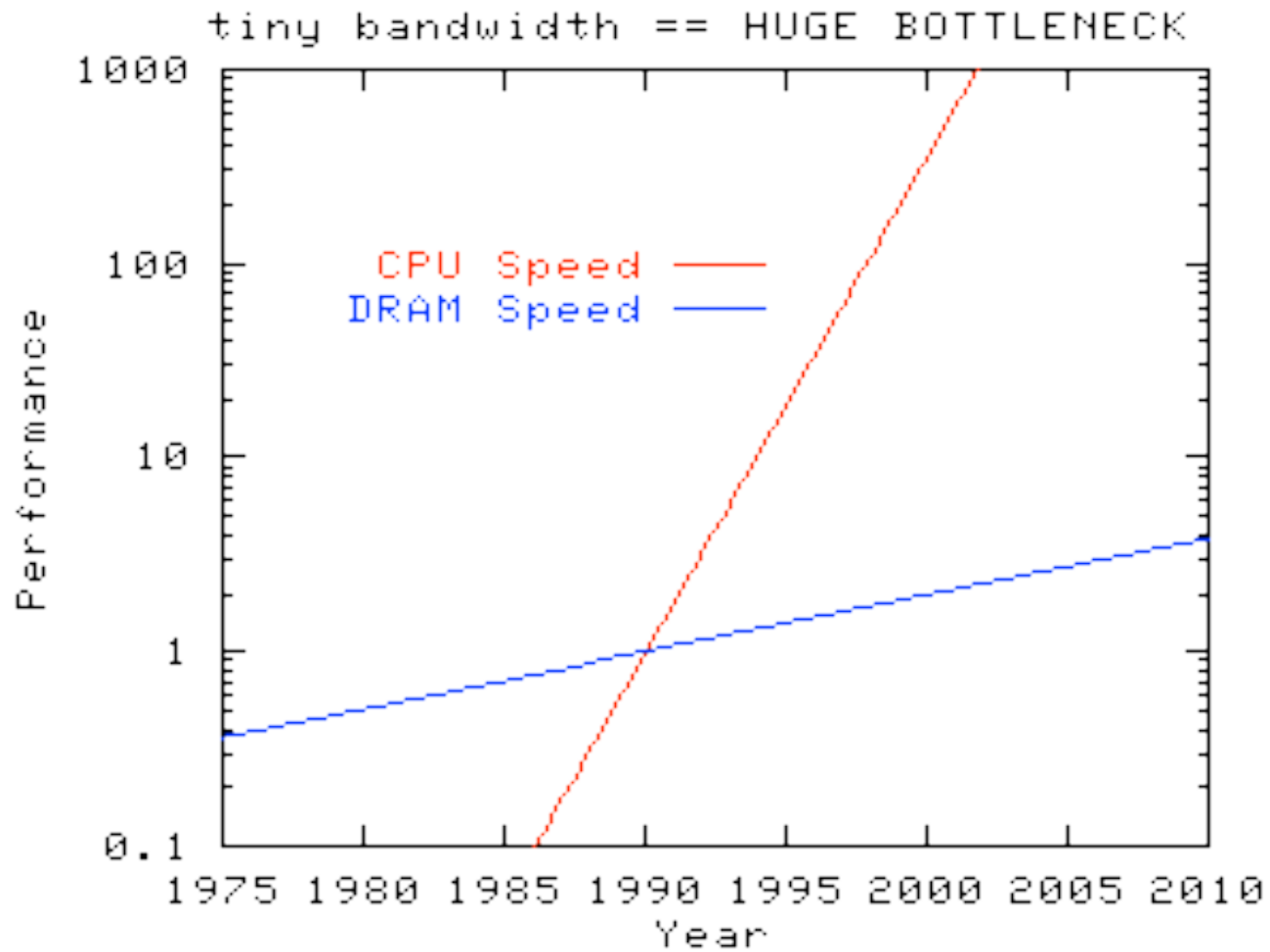
MAIN MEMORY

COMPUTATION

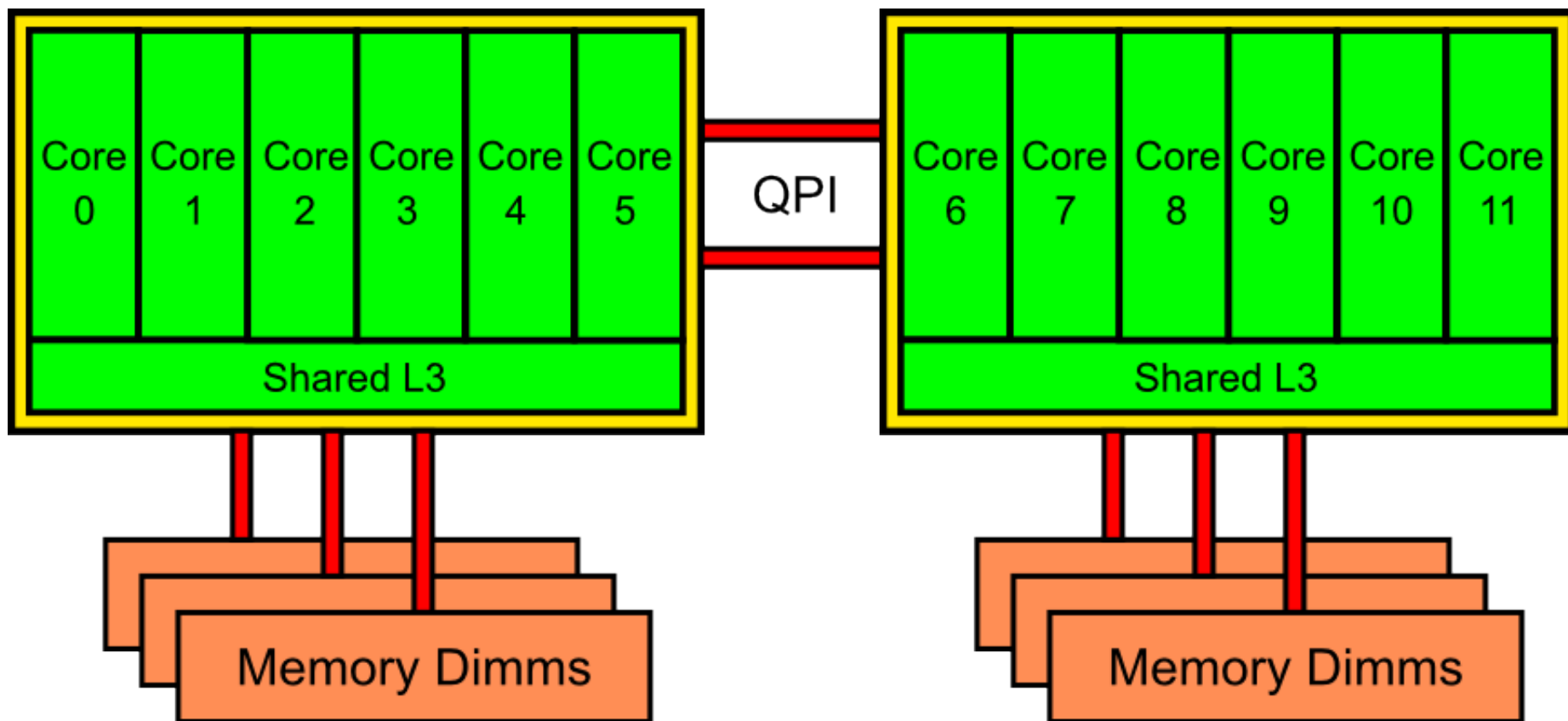
APPLICATION DATA

# The CPU Memory Hierarchy



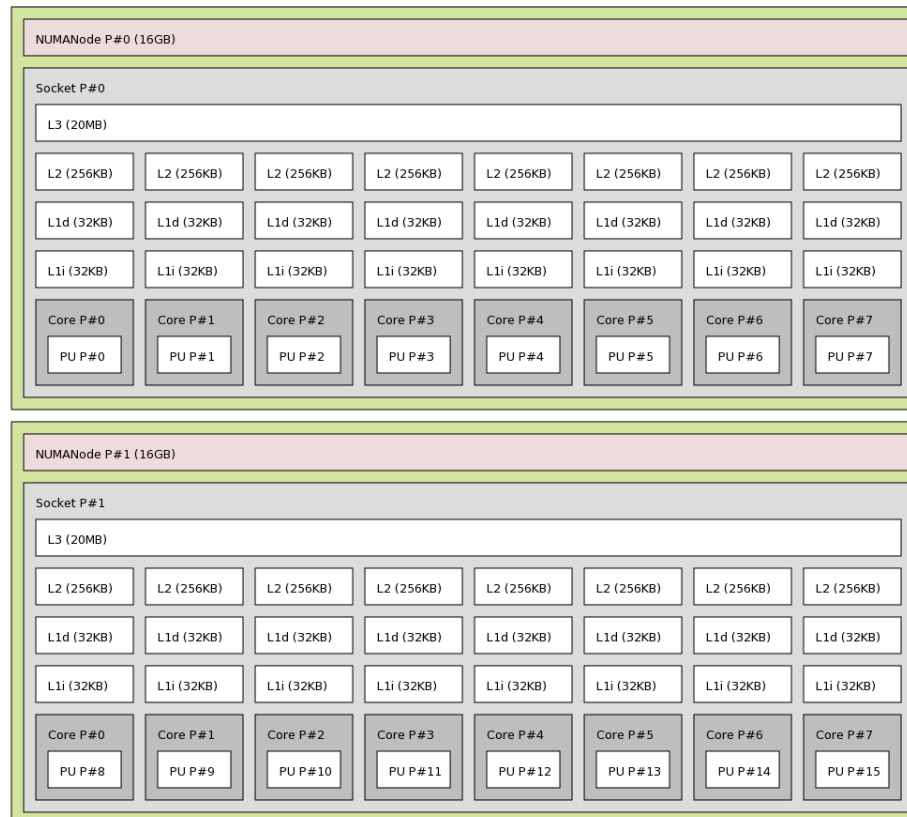


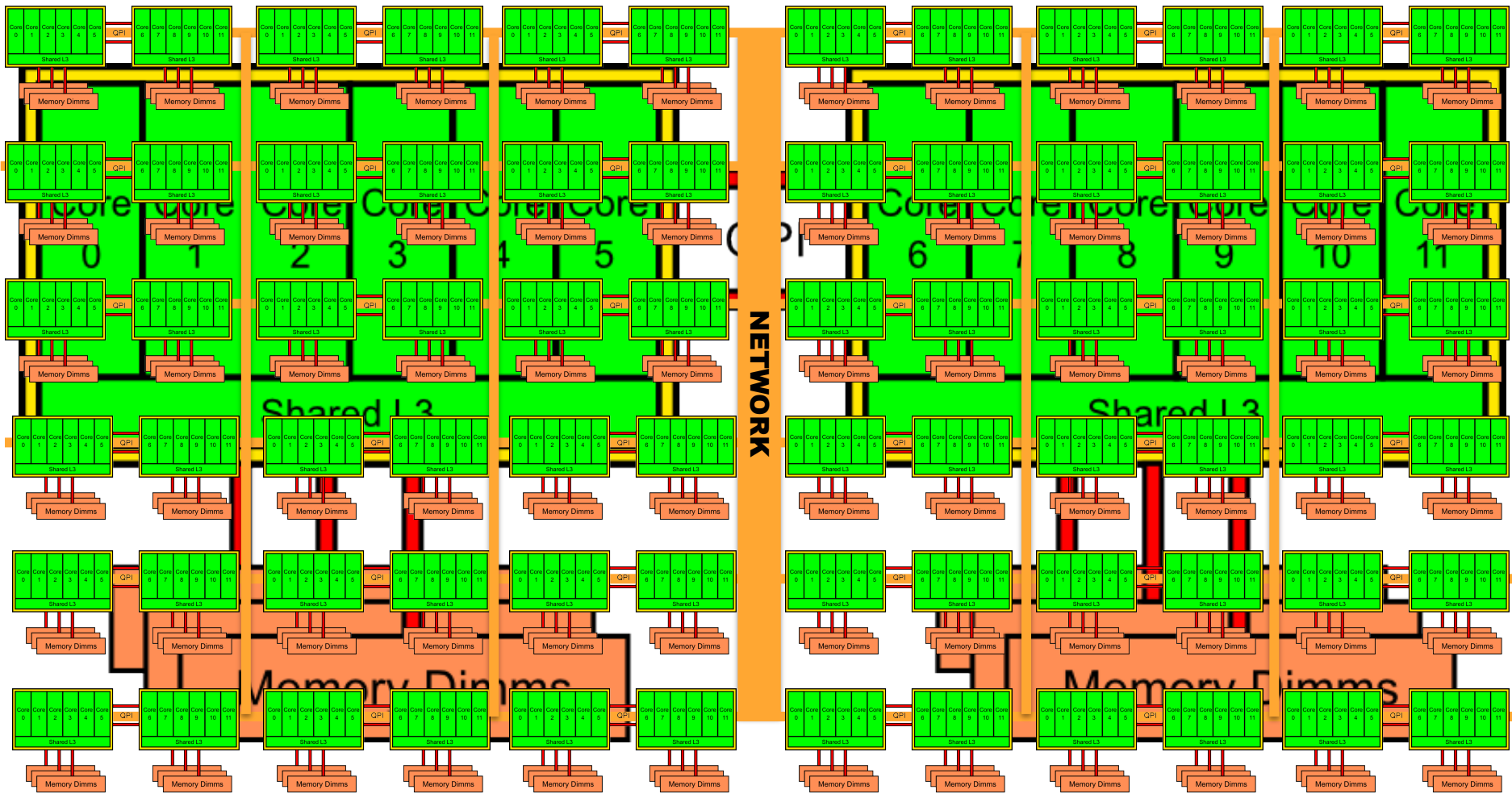
# Multiple Socket CPUs



# Intel Xeon E5-2665 Sandy Bridge-EP 2.4GHz

Machine (32GB)







The Abdus Salam  
International Centre  
for Theoretical Physics



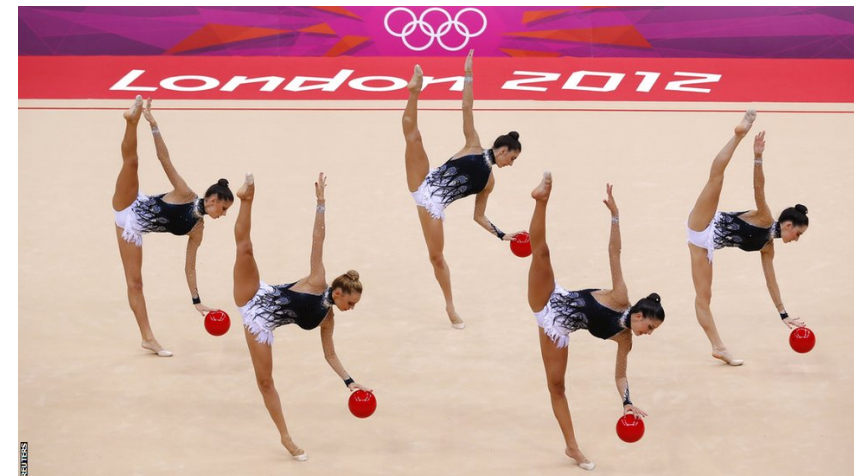
# What Determines Performance?

- How fast is my CPU?
- How fast can I move data around?
- How well can I split work into pieces?
  - Very application specific: never assume that a good solution for one problem is as good a solution for another
  - always run benchmarks to understand requirements of your applications and properties of your hardware
  - respect Amdahl's law

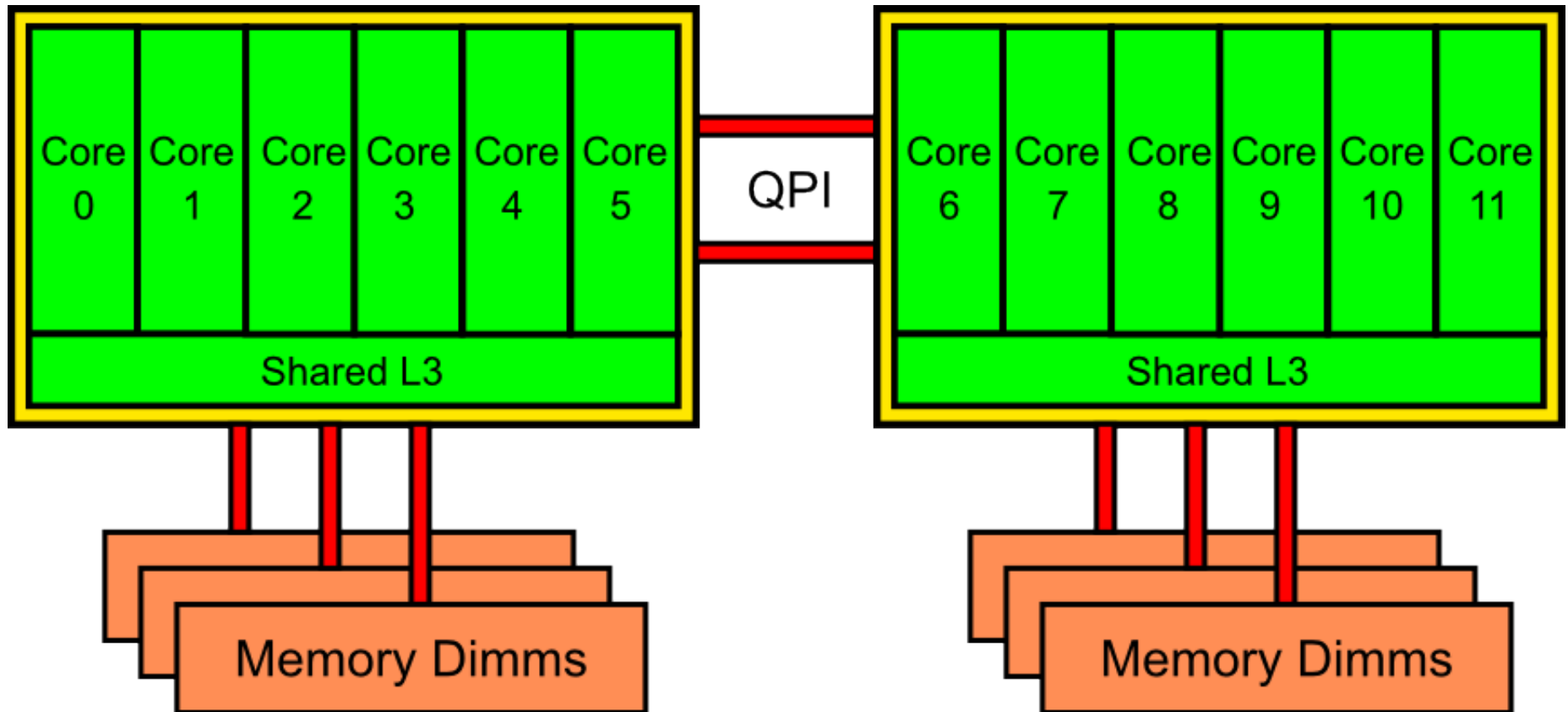


# Type of Parallelism

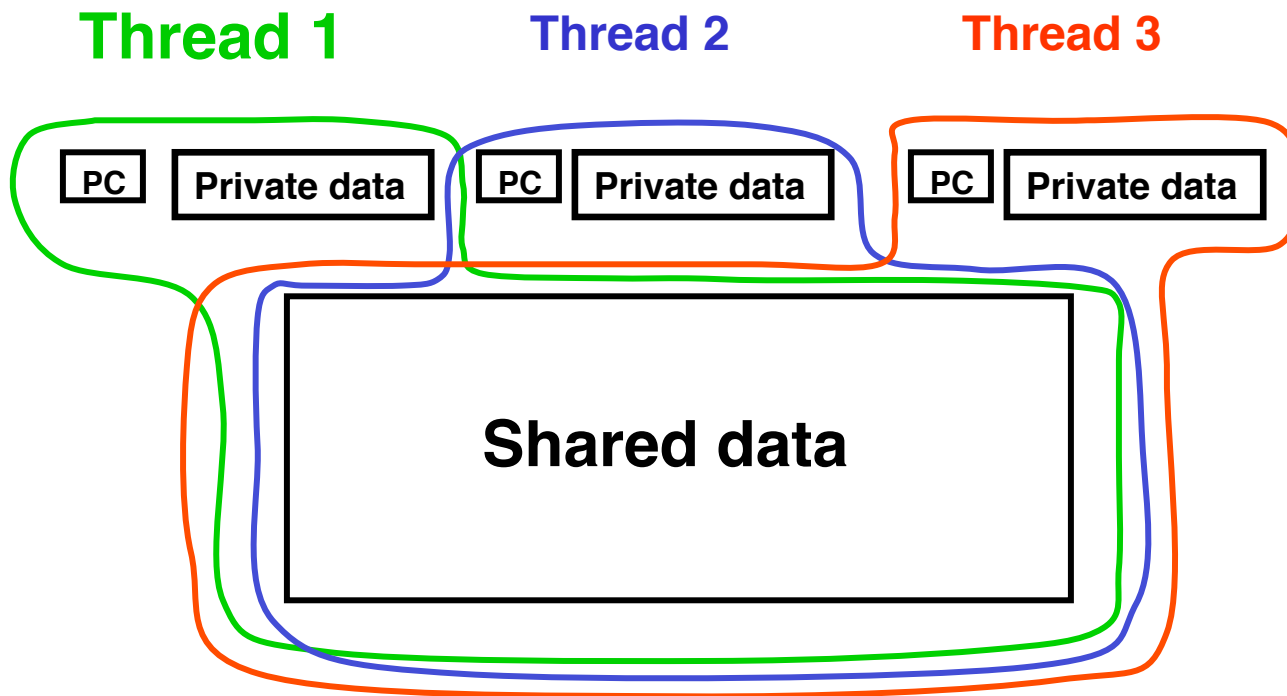
- **Functional (or task) parallelism:**  
different people are performing different task at the same time
- **Data Parallelism:**  
different people are performing the same task, but on different equivalent and independent objects



# Multiple Socket CPUs



# Paradigm at Shared Memory /1





# Paradigm at Shared Memory /2

- Usually indicated as Multithreading Programming
- Commonly implemented in scientific computing using the OpenMP standard (directive based)
- Thread management overhead
- Limited scalability
- Write access to shared data can easily lead to race conditions and incorrect data



# Parallel Programming Paradigms

- MPI (Message Passing Interface)
  - A standard defined for portable message passing
  - It available in the form of library which includes interfaces for expressing the data exchange among processes
  - A framework is provided for spawning the independent processes (i.e., mpirun)
  - Processes communication is via network
  - It works on either shared and distributed mem. architecture
  - ideal for distributing memory among compute nodes

# MPI Program Design

- Multiple and separate processes (can be local and remote) concurrently that are coordinated and exchange data through “messages” => a “share nothing” parallelization
- Best for coarse grained parallelization Distribute large data sets; replicate small data
- Minimize communication or overlap communication and computing for efficiency => Amdahl's law



# What is MPI?

- A standard, i.e. there is a document describing how the API (constants & subroutines) are named and should behave; multiple “levels”, **MPI-1** (basic), MPI-2 (advanced), MPI-3 (new)
- A library or API to hide the details of low-level communication hardware and how to use it
- Implemented by multiple vendors
- Open source and commercial versions
- Vendor specific versions for certain hardware
- Not binary compatible between implementations

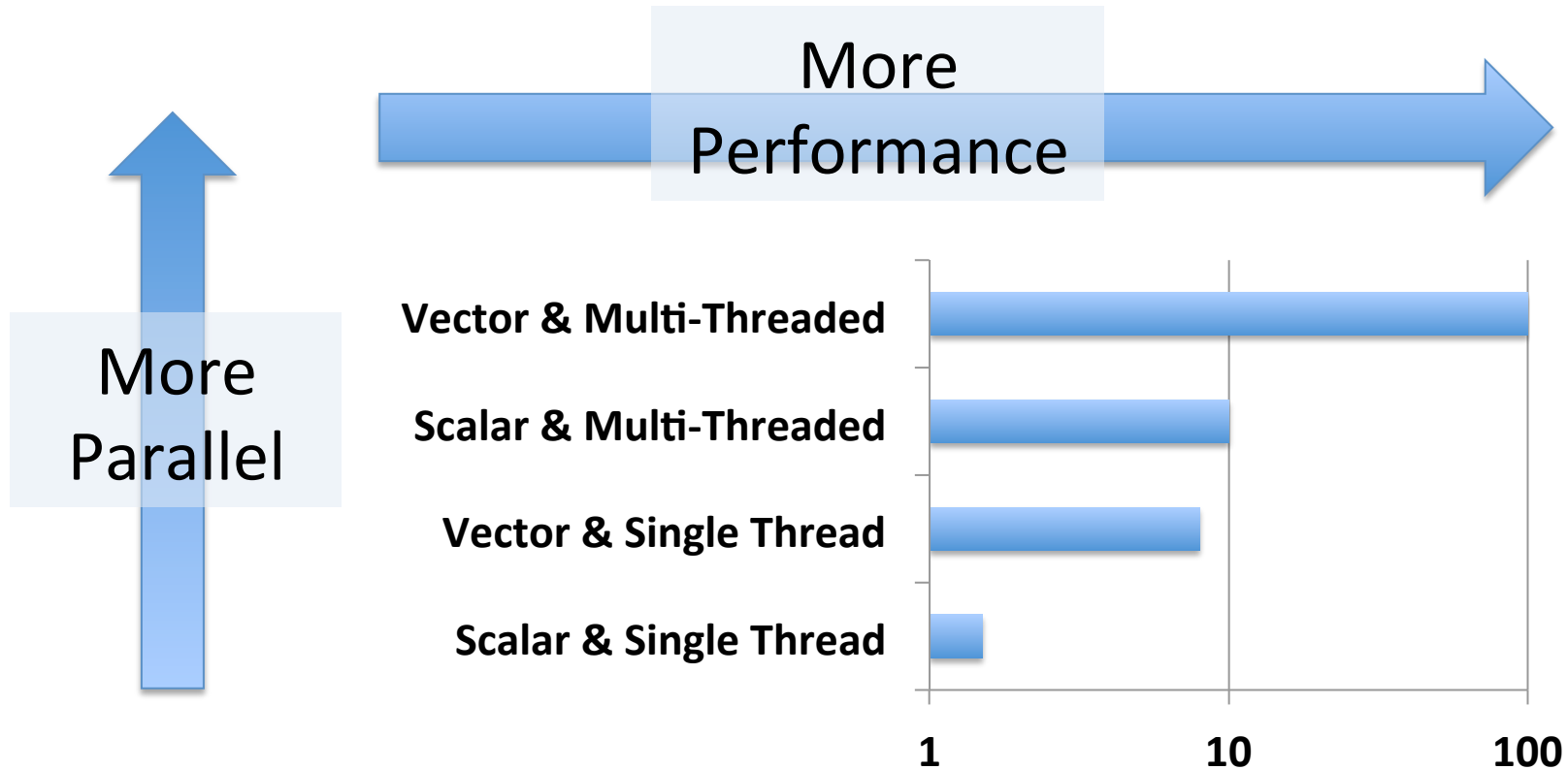
# Programming Parallel Paradigms

- Are the tools we use to express the parallelism for on a given architecture
- They differ in how programmers can manage and define key features like:
  - parallel regions
  - concurrency
  - process communication
  - synchronism



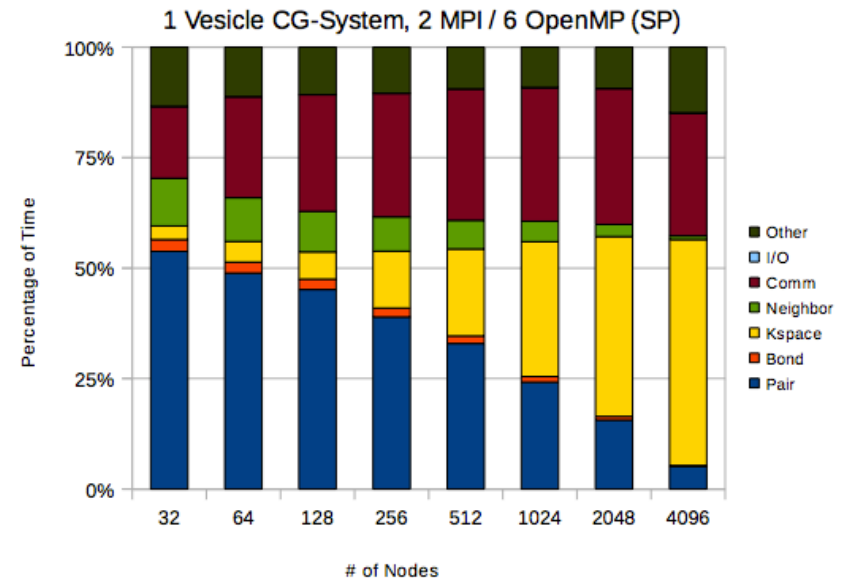
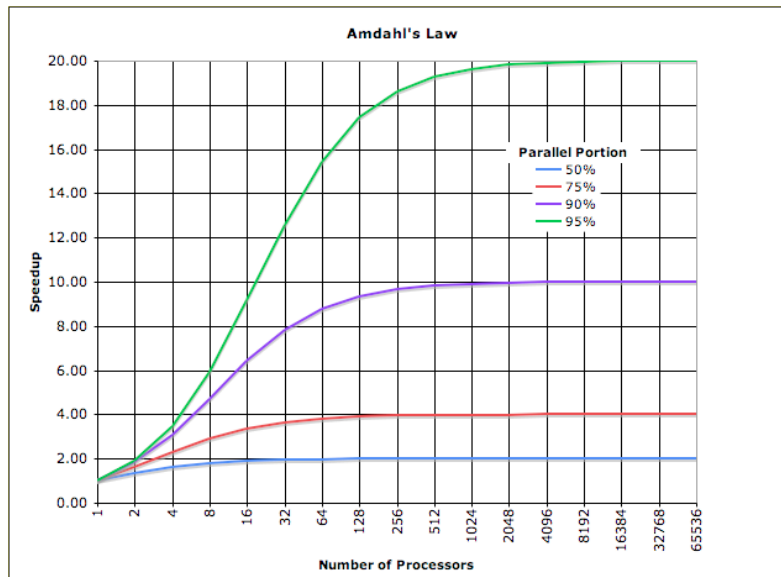


# Threading and Vectorization

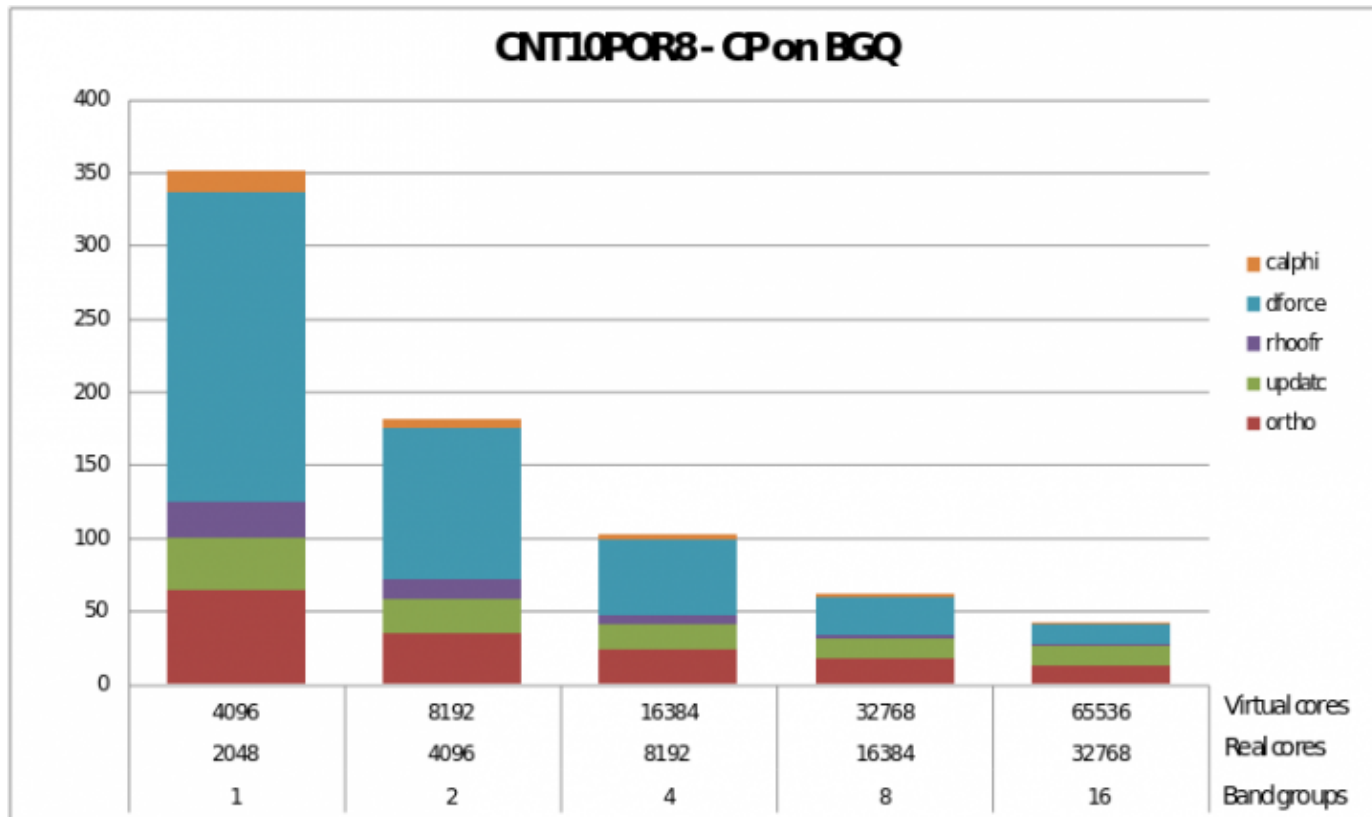


# Amdal's Law And Real Life

- The speedup of a parallel program is limited by the sequential fraction of the program
- This assumes perfect scaling and no overhead



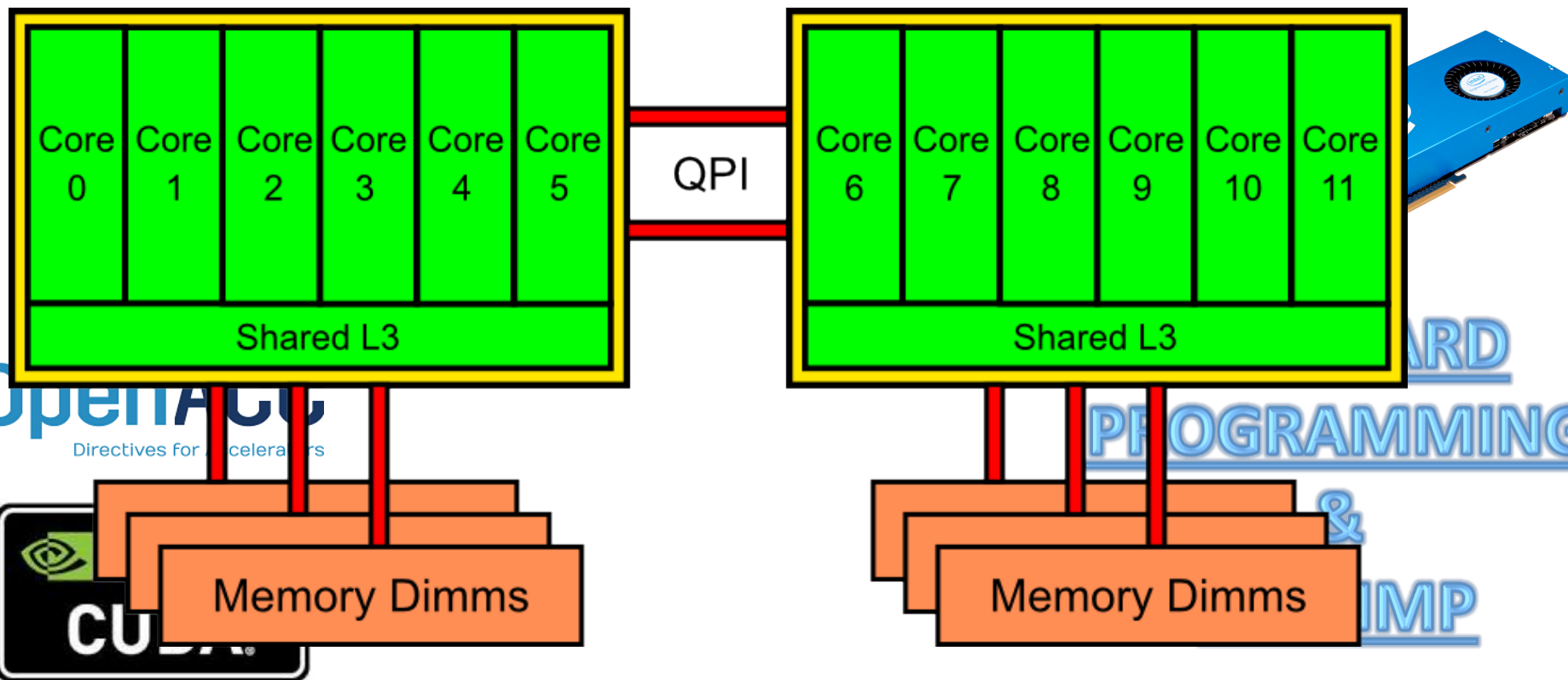
# Scaling - QE-CP on Fermi BGQ @ CINECA



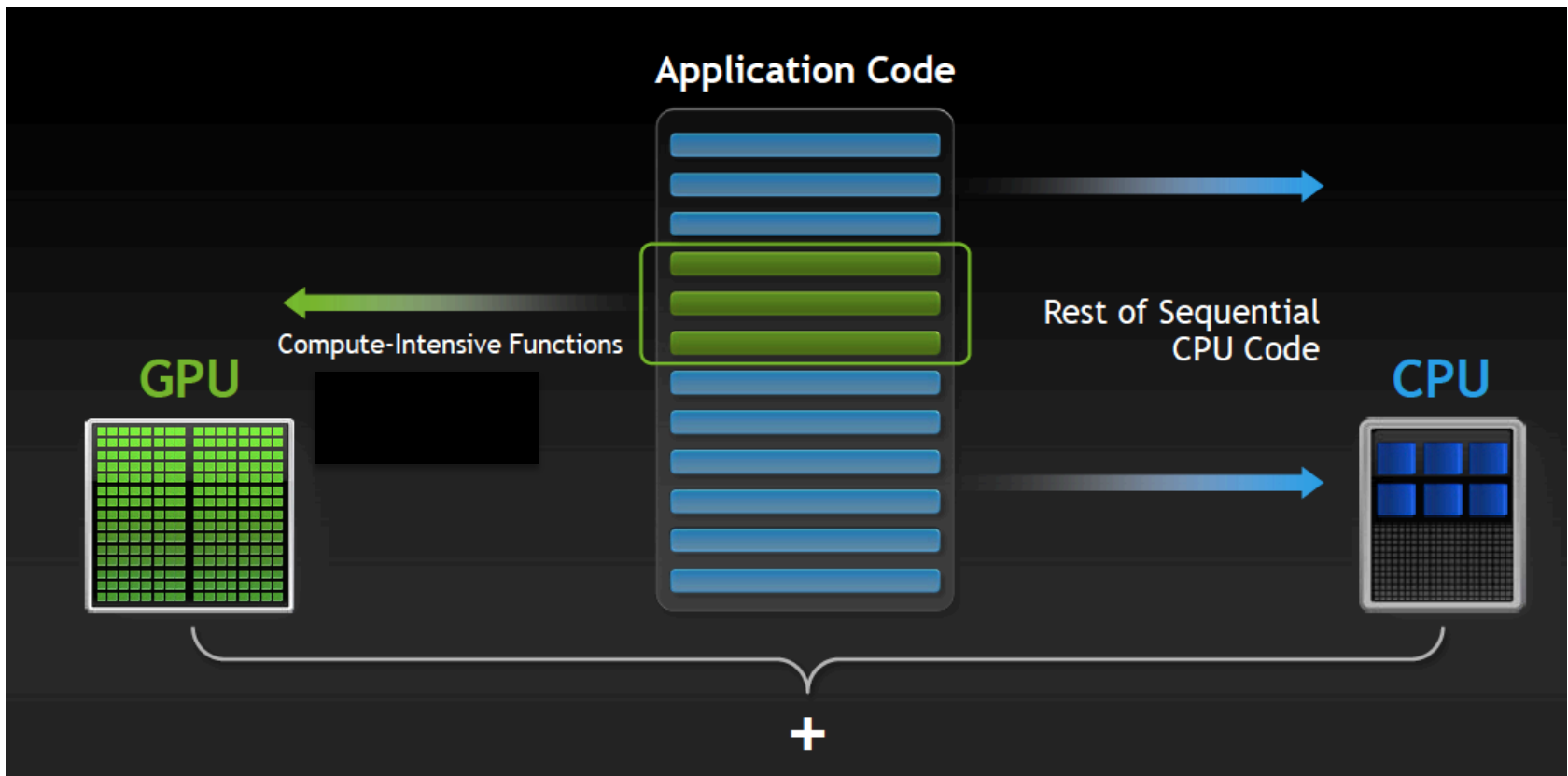
# Easy Parallel Computing

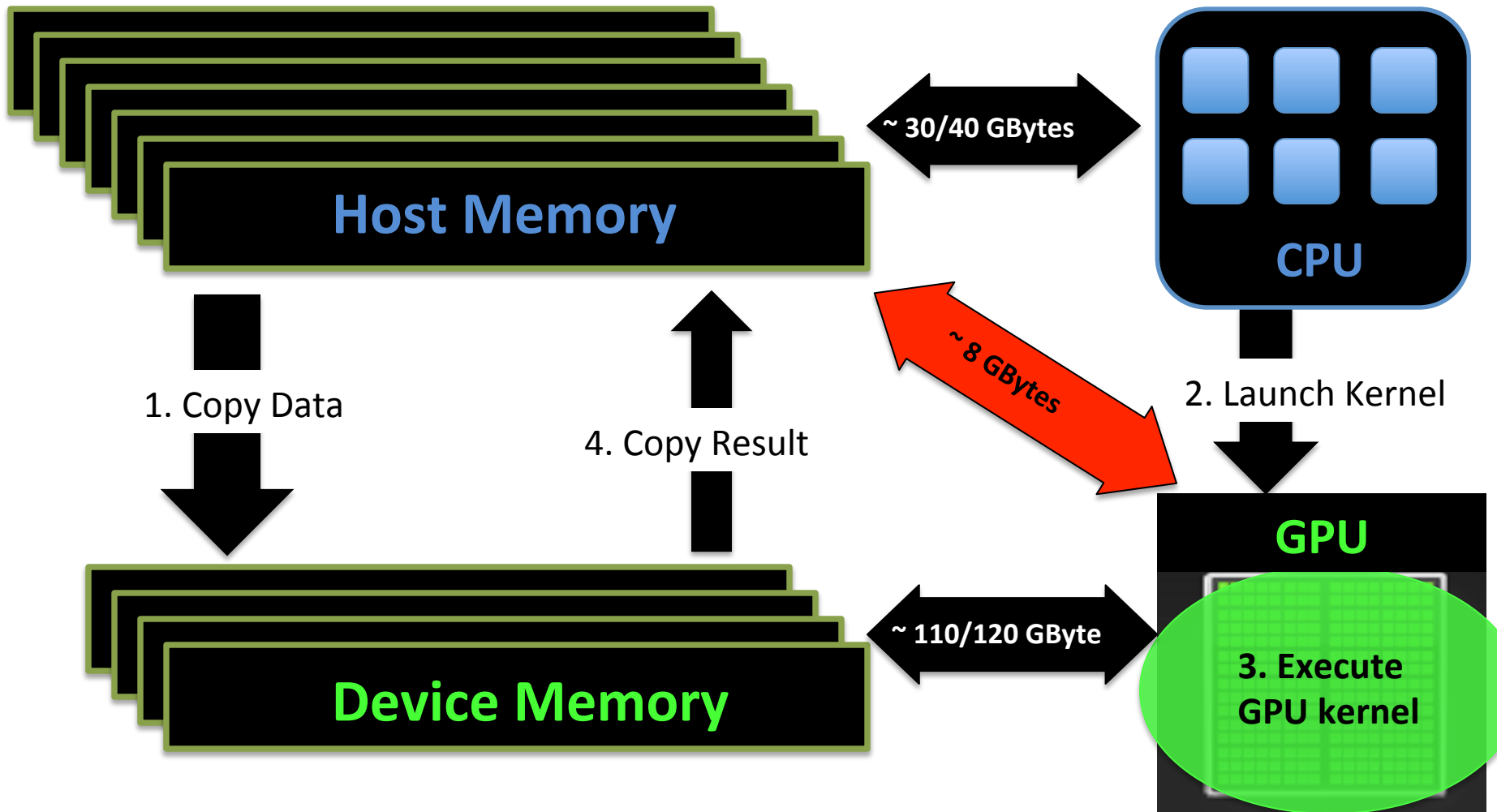
- Farming, embarrassingly parallel
  - Executing multiple instances on the same program with different inputs/initial cond.
  - Reading large binary files by splitting the workload among processes
  - Searching elements on large data-sets
  - Other parallel execution of embarrassingly parallel problem (no communication among tasks)
- Ensemble simulations (weather forecast)
- Parameter space (find the best wing shape)

# Multiple Socket CPUs + Accelerators



# The General Concept of Accelerated Computing







# Why Does GPU Accelerate Computing?

- Highly scalable design
- Higher aggregate memory bandwidth
- Huge number of low frequency cores
- Higher aggregate computational power
- Massively parallel processors for data processing



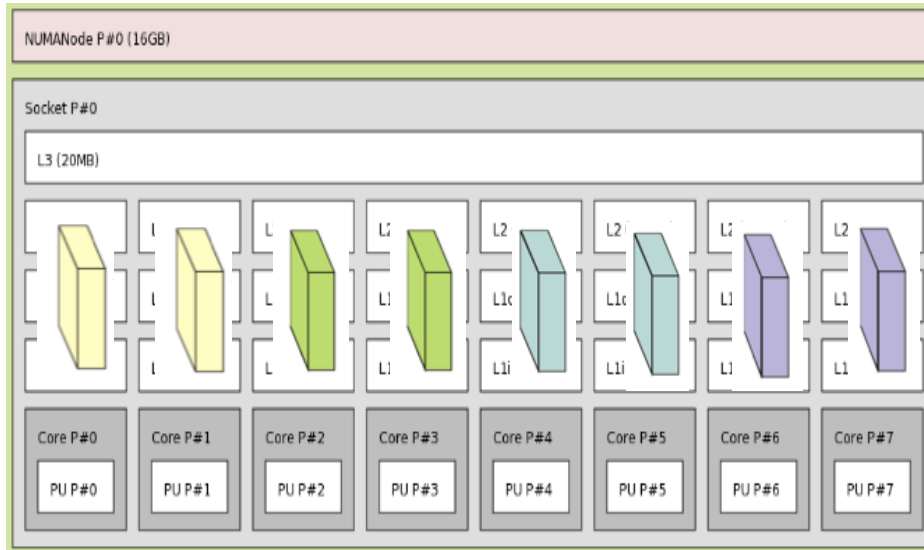
# Why Does GPU Not Accelerate Computing?

- PCI Bus bottleneck
- Synchronization weakness
- Extremely slow serialized execution
- High complexity
  - SPMD(T) + SIMD & Memory Model
- People forget about the Amdahl's law
  - accelerating only the 50% of the original code, the expected speedup can get at most a value of 2!!



# Higher aggregate computational power

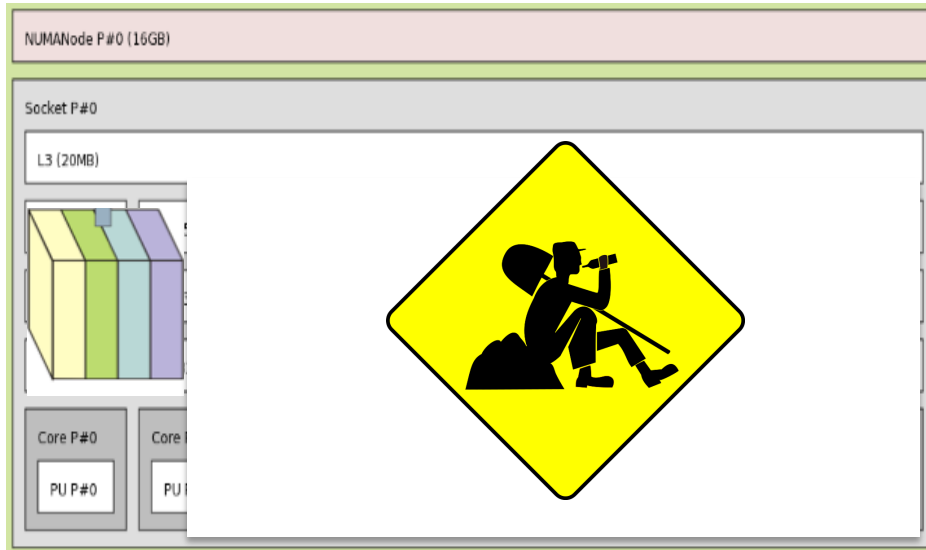
- Do we really ... need it? ... have it available?
- Can we really exploit it?
- Remember the key-factors for performance
  - #operations per clock cycle x frequency x #cores
  - the DP power is drastically reduced if the compute capability is only partially exploited
- How much is my GPU better than my CPU?
- Can data move from CPU2GPU and from GPU2CPU be reduced?
- For general purpose and scalable applications, both CPU and GPU must usually be exploited



The Intel Xeon E5-2665  
Sandy Bridge-EP 2.4GHz



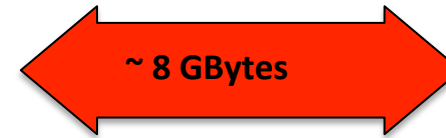
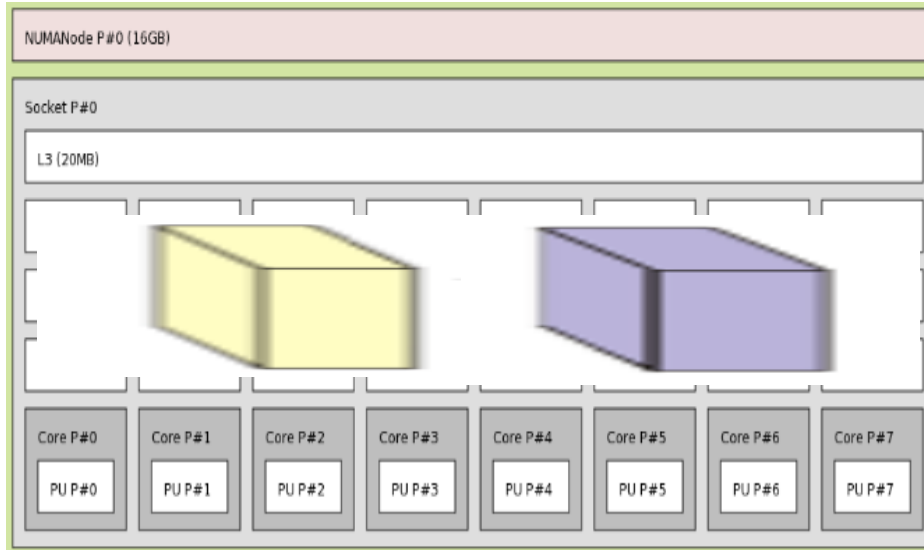
```
mpirun -np 8 pw-gpu.x -inp input file
```



The Intel Xeon E5-2665  
Sandy Bridge-EP 2.4GHz



```
mpirun -np 1 pw-gpu.x -inp input file
```



The Intel Xeon E5-2665  
Sandy Bridge-EP 2.4GHz

```
export OMP_NUM_THREADS=4
export OPENBLAS_NUM_THREADS=$OMP_NUM_THREADS
mpirun -np 2 pw-gpu.x -inp input file
```

Workload Management: system level, High-throughput

Python: Ensemble simulations, workflows

MPI: Domain partition

OpenMP: Node Level shared mem

CUDA/OpenCL/OpenAcc:  
floating point accelerators

# Summary

- The memory bandwidth is one of the most critical bottleneck of the modern CPU design, specially considering at the increasing of the number of cores
- Efficient access data patterns and process/memory affinity are inescapable for decent performances
- Exploitation of the different levels of parallelism is needed to push at the limit the efficiency of the CPU:
  - superscalar pipelining, FMA, SIMD, multicores
- Complexity of parallel systems requires a given technological background to master the load-balancing (also at user level) and obtain a decent efficiency

# Consequence on Software Applications

- Moore's law continues, but leads to multi-core, larger caches and higher integration => Performance increase now mostly through better algorithms, optimization, vectorization, and parallelization
- Bottleneck has transitioned from CPU speed to memory access and efficient data structures
- Write cleaned and structured code to enhance the compiler and perform possible optimizations
  - Use the compiler optimizations: -O3, -msse4, -mAVX, etc ... !!!
- Best optimization requires often a writing/rewriting for reducing the complexity and/or help the compiler to work efficiently
- Make use of optimized software (i.e., libraries)
- High availability of massive compute power drives to more complex SW



# Libraries



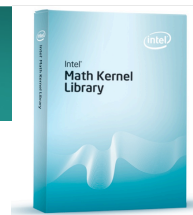
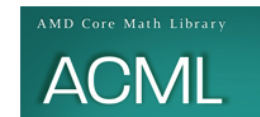
Highly-portable  
(included into some  
Linux distributions)



Freely available  
Open Source Optimized



Third-Party Highly-Optimized





## What If You Want to Learning How to Program All This?!

- Introductory School on Parallel Programming and Parallel Architecture for High Performance Computing | (smr 2877)
- 3 October 2016 - 14 October 2016

## What If You Want to Master All This?!



# MHPC

Master in High Performance Computing



- ABOUT
- OBJECTIVES
- COURSES
- APPLY
- FAQ
- PEOPLE
- SPONSORS



### MHPC

The Master in High-Performance Computing (MHPC) is a high-level degree program that aims to train students to solve complex problems with HPC techniques.

### WHY

Set in a stimulating research environment, the MHPC is an innovative, hands-on training and education program to prepare students for exciting careers in the fast-growing field of HPC.

### THE TARGET

The master is intended for people with strong interest in advanced programming for scientific computing, software optimization and management of computing platforms.

[READ ALL](#)



The Abdus Salam  
International Centre  
for Theoretical Physics



IAEA  
International Atomic Energy Agency

# Thanks for your attention!!





The Abdus Salam  
International Centre  
for Theoretical Physics



# Hands On

**Ivan Girotto – [igirotto@ictp.it](mailto:igirotto@ictp.it)**

Information & Communication Technology Section (ICTS)  
International Centre for Theoretical Physics (ICTP)