The Abdus Salam International Centre for Theoretical Physics

Foundation of Computer Architecture for HPC

Ivan Girotto – 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

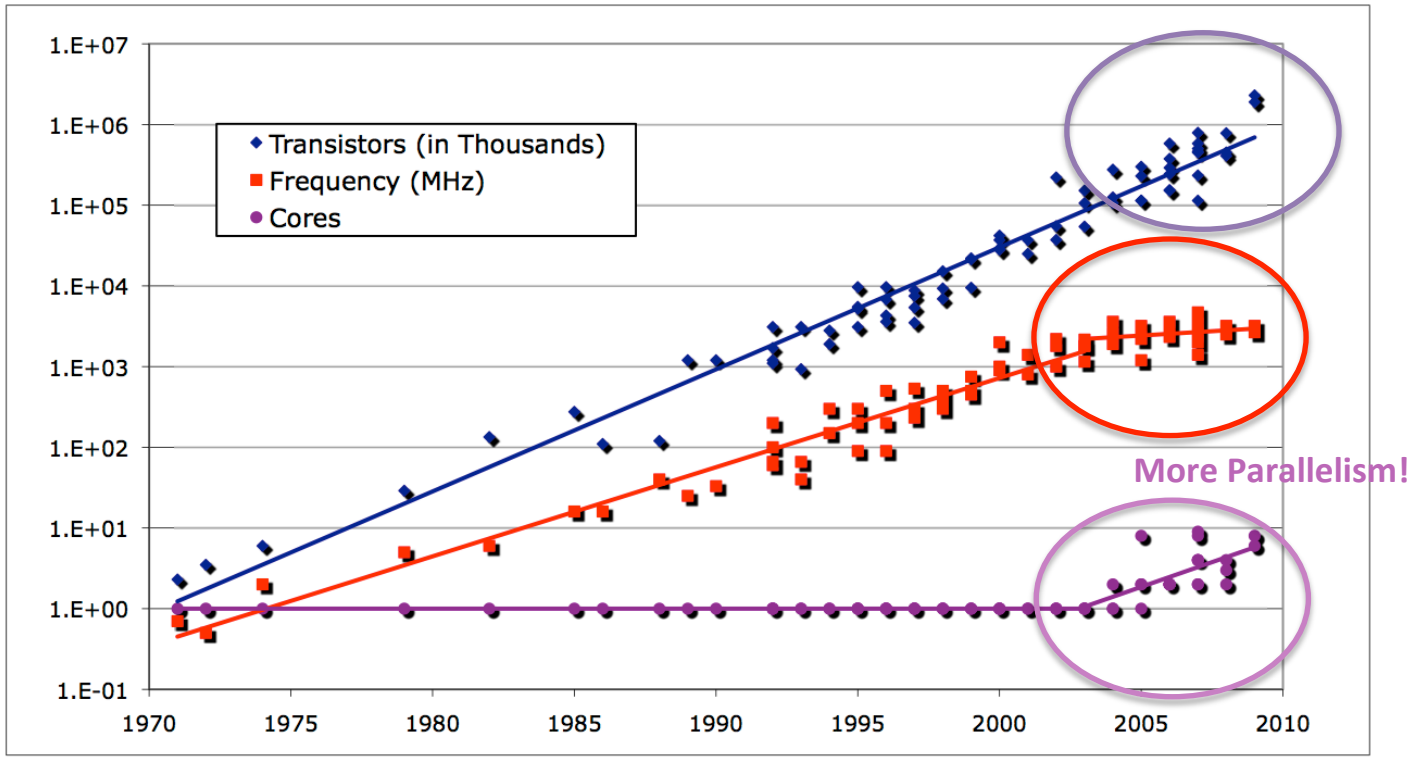
Why would HPC matter to you?

- 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

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

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

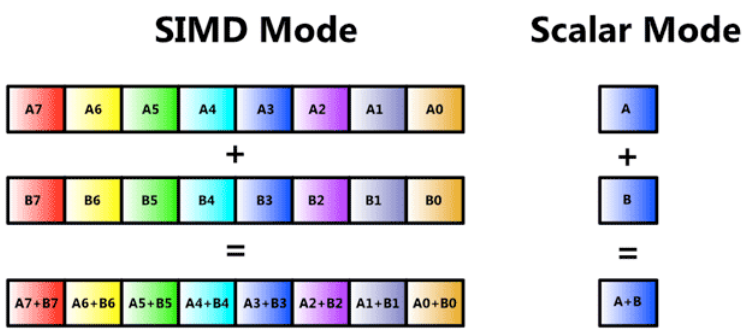
ARM → Main focus on low power mobile chip
Qualcomm, Texas inst. , Nvidia, ST, ecc
new HPC market, server maket

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

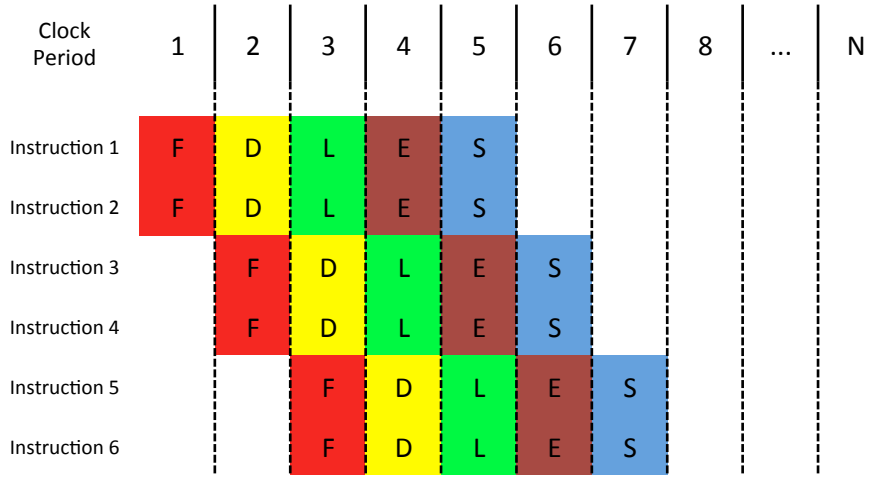
IBM → Embedded market
Power+GPU, only chance for HPC

AMD → Console market
Still some chance for HPC

To the Extreme - Parallel Inside



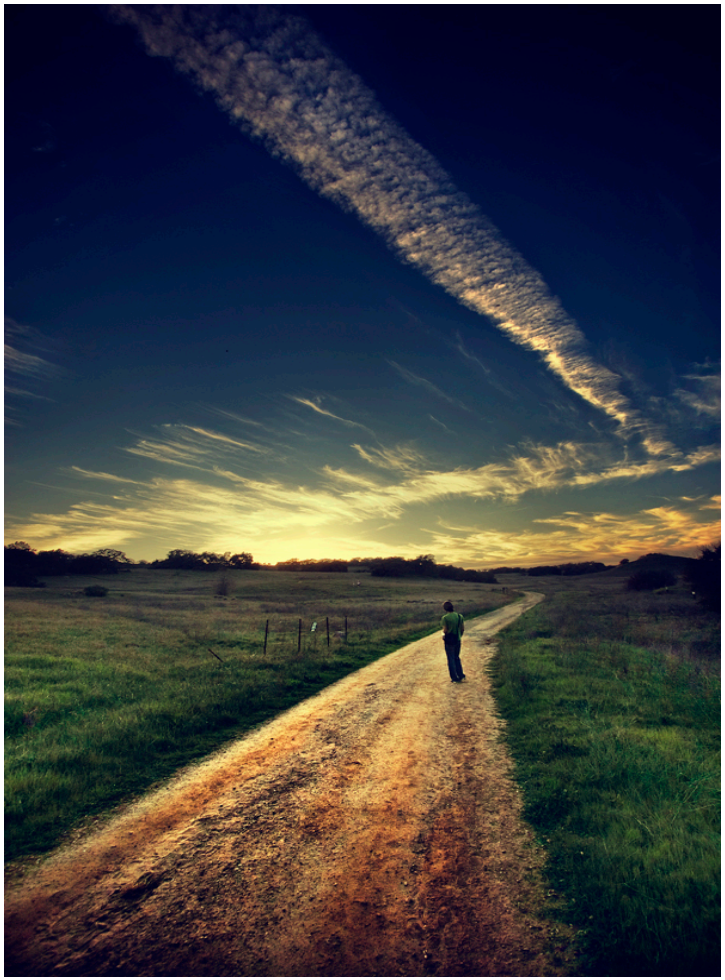
Vector Units for processing multiple data in //



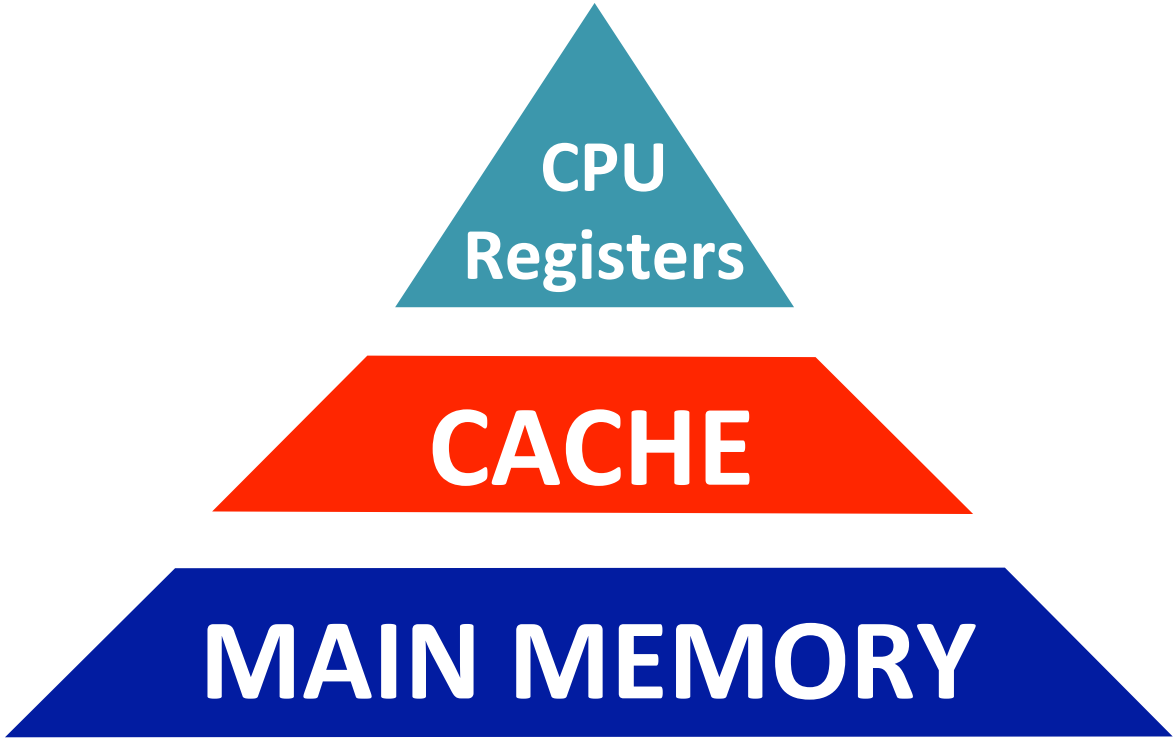
Pipelined/Superscalar design: multiple functional units operate concurrently

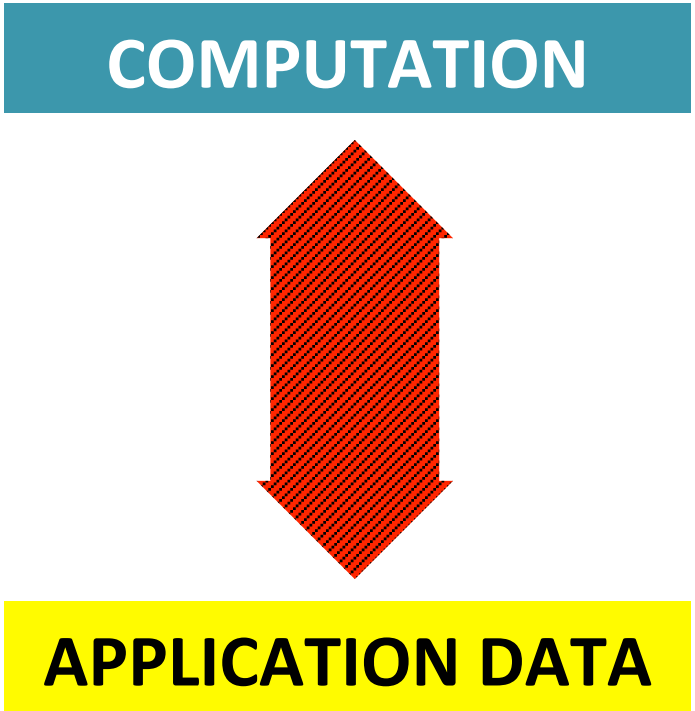
Consequences

- 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

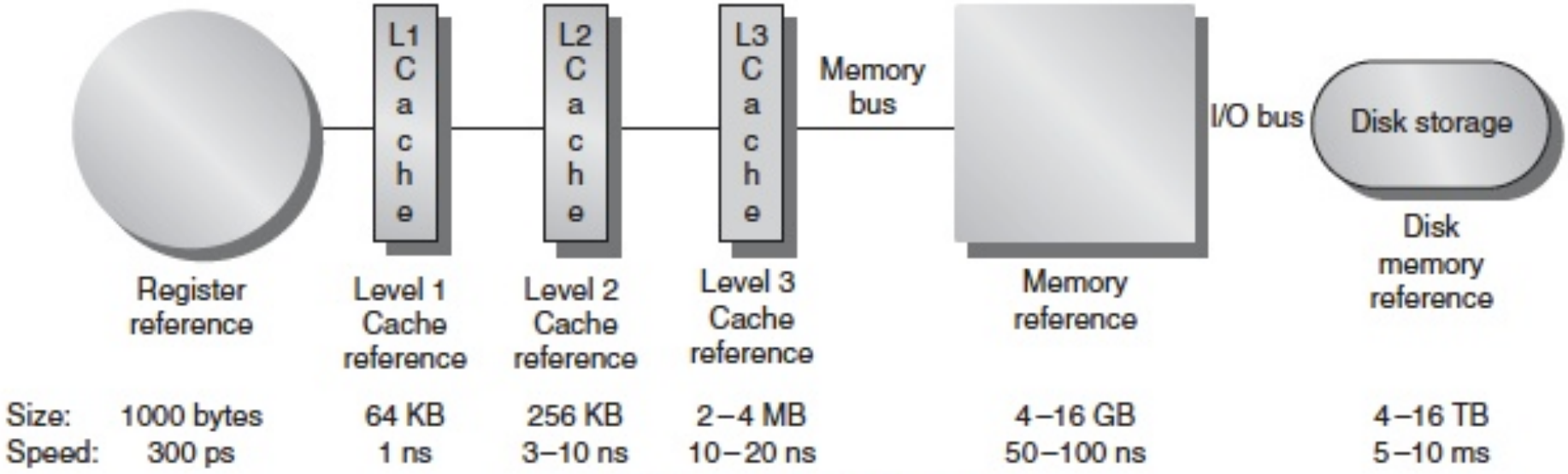


The CPU Memory Hierarchy

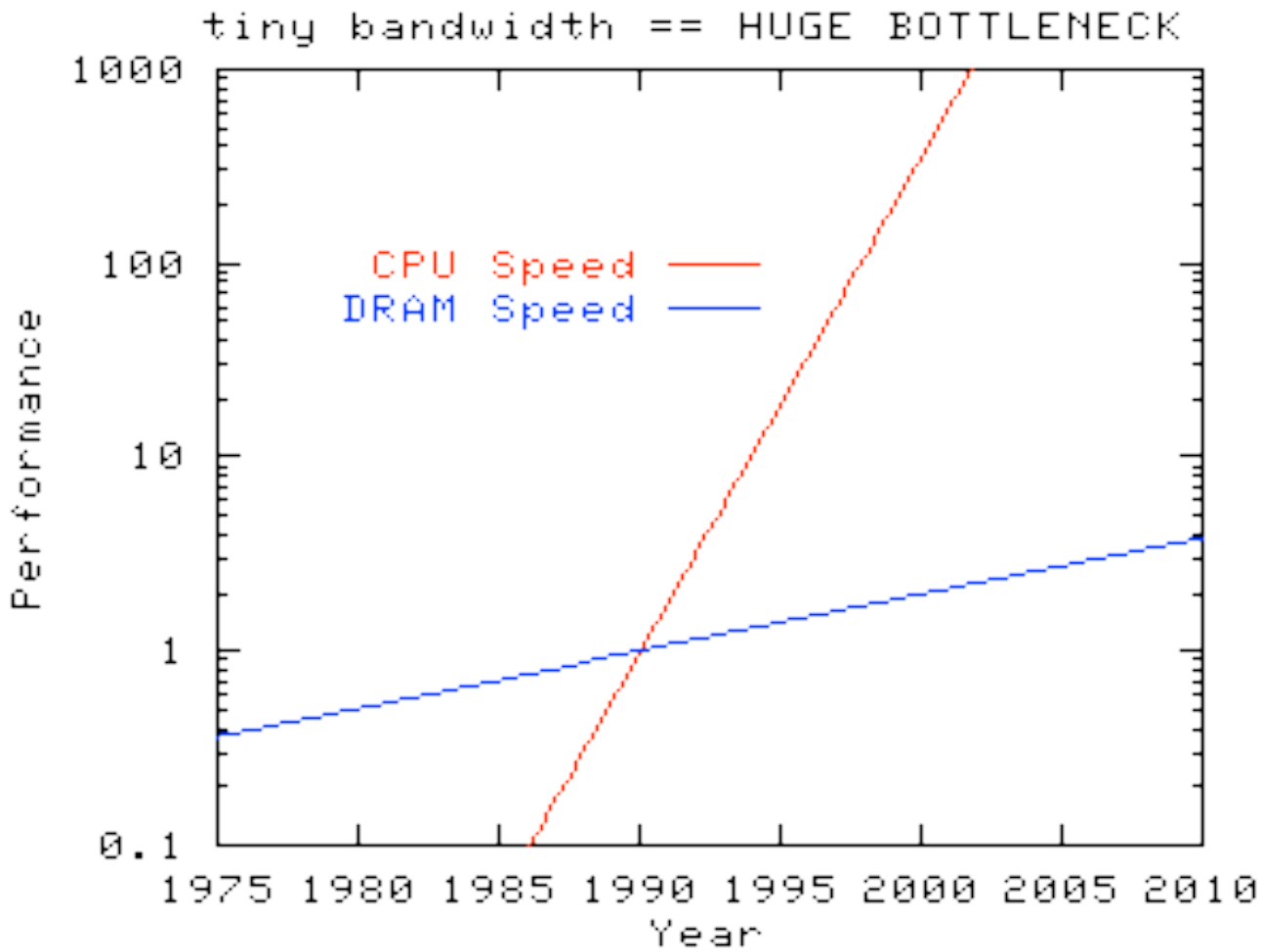




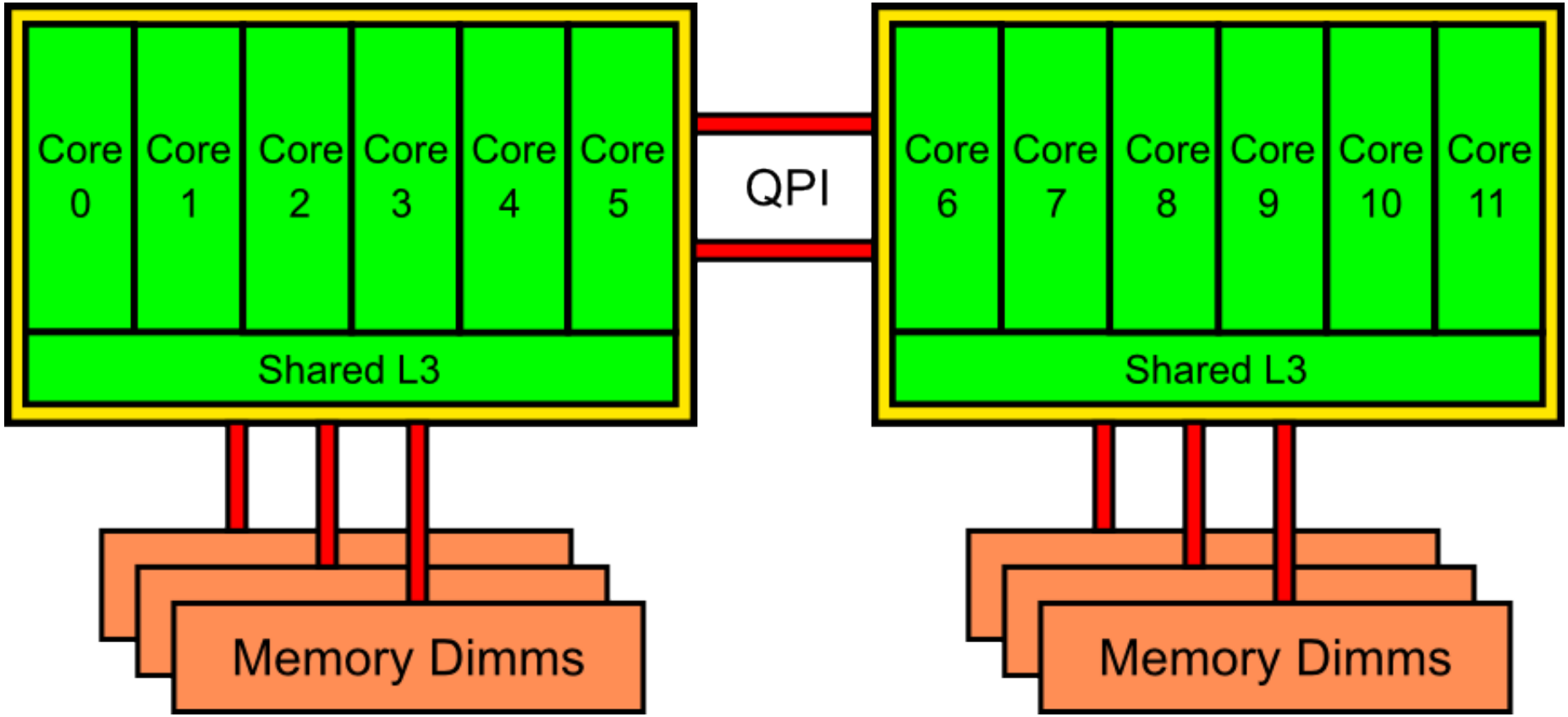
The CPU Memory Hierarchy



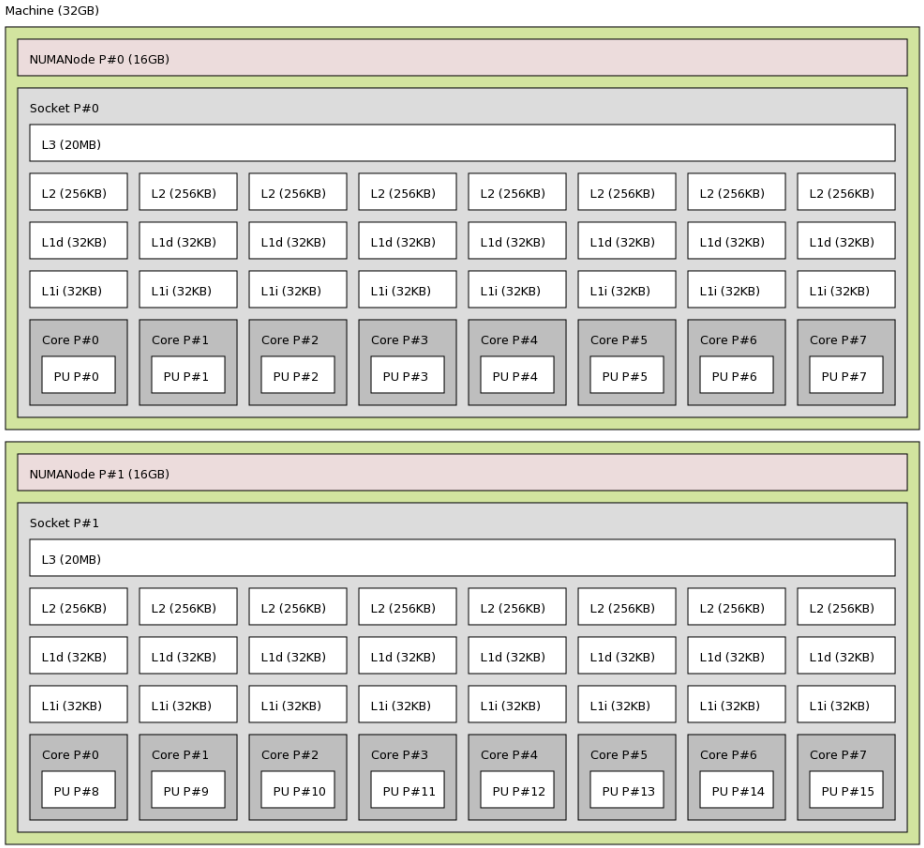
(a) Memory hierarchy for server

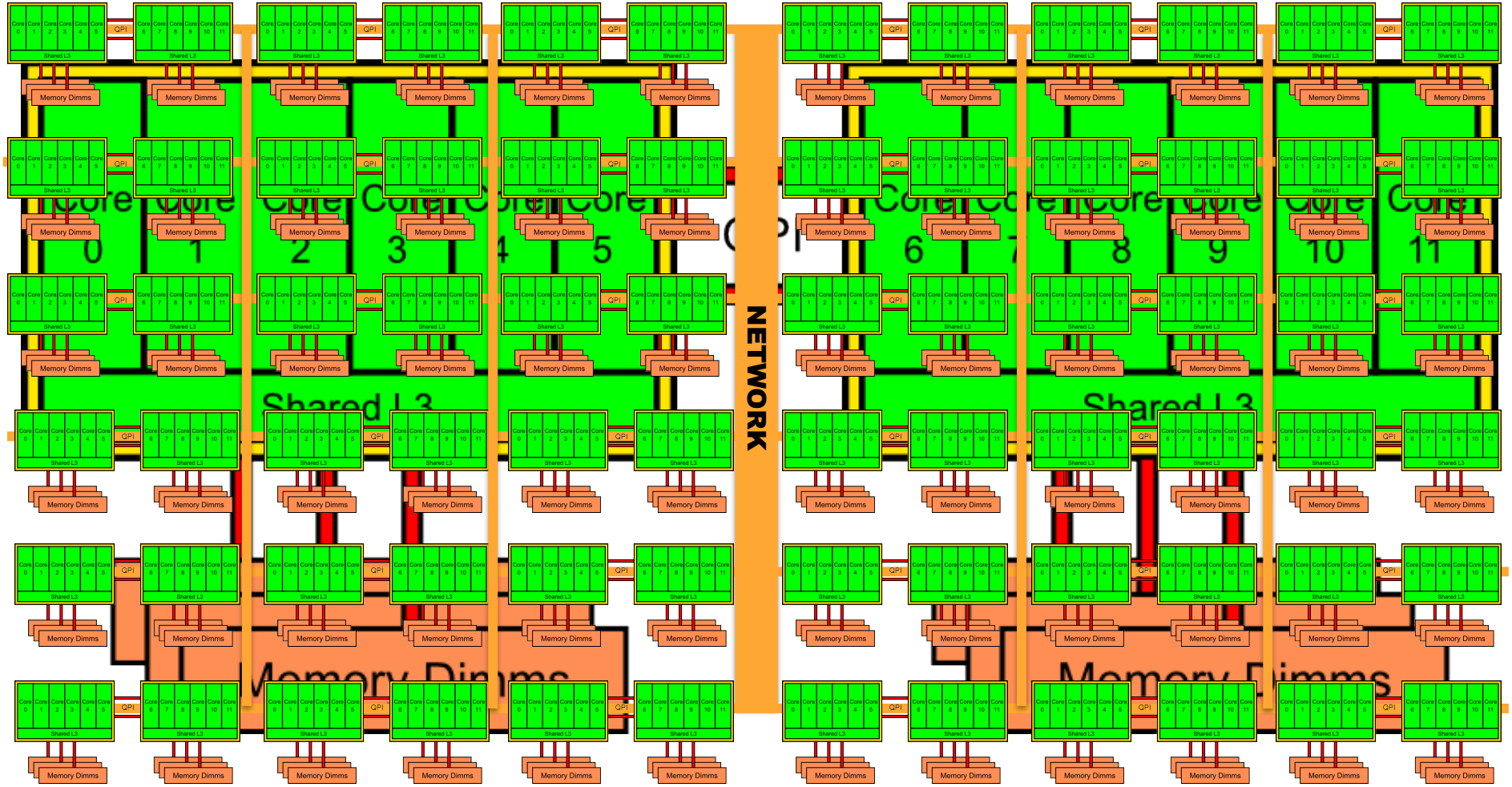

tiny bandwidth == HUGE BOTTLENECK
1000
100
10
1
0.1
Performance
CPU Speed
DRAM Speed
1975 1980 1985 1990 1995 2000 2005 2010
Year

Multiple Socket CPUs

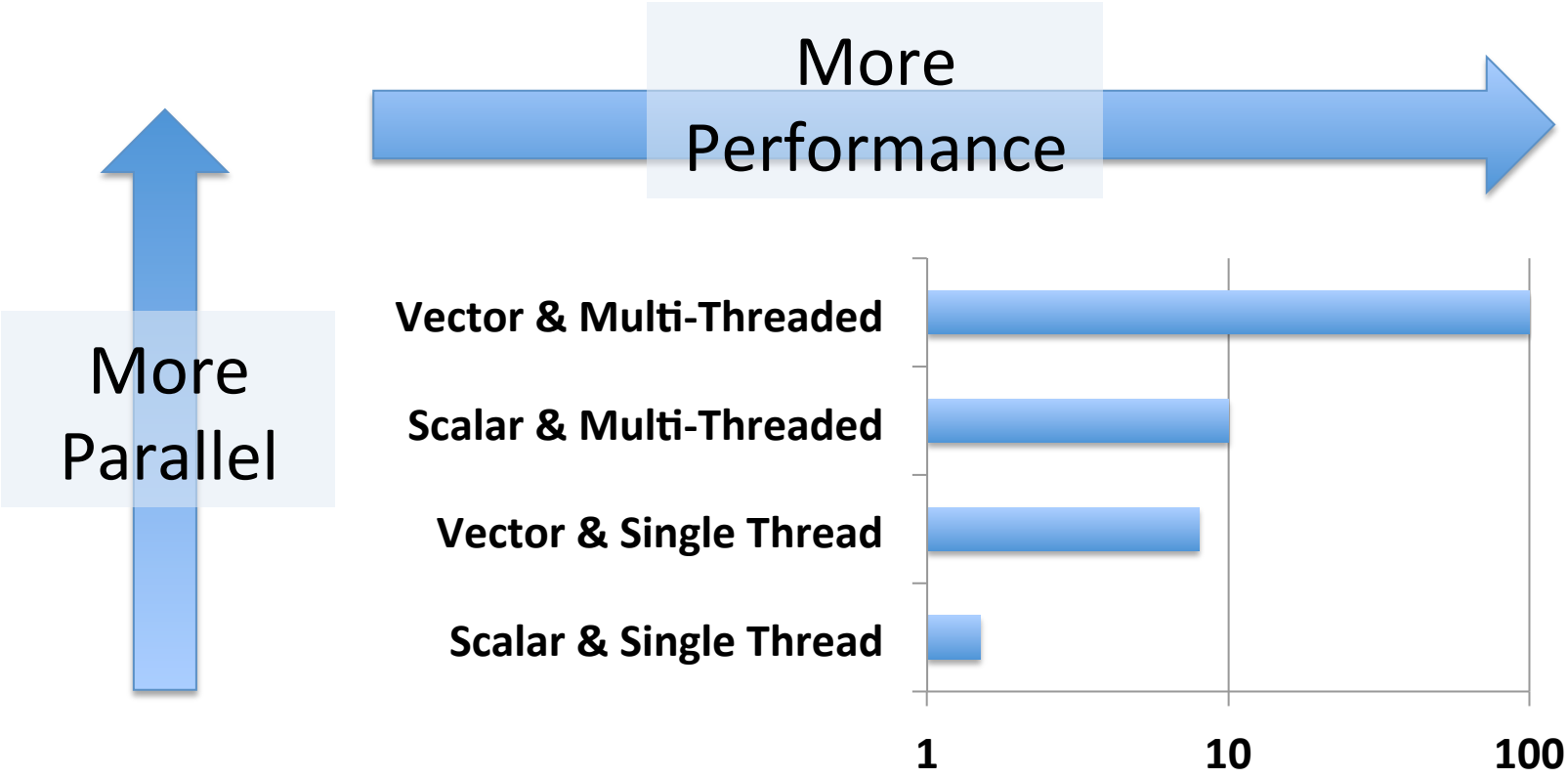


Intel Xeon E5-2665 Sandy Bridge-EP 2.4GHz

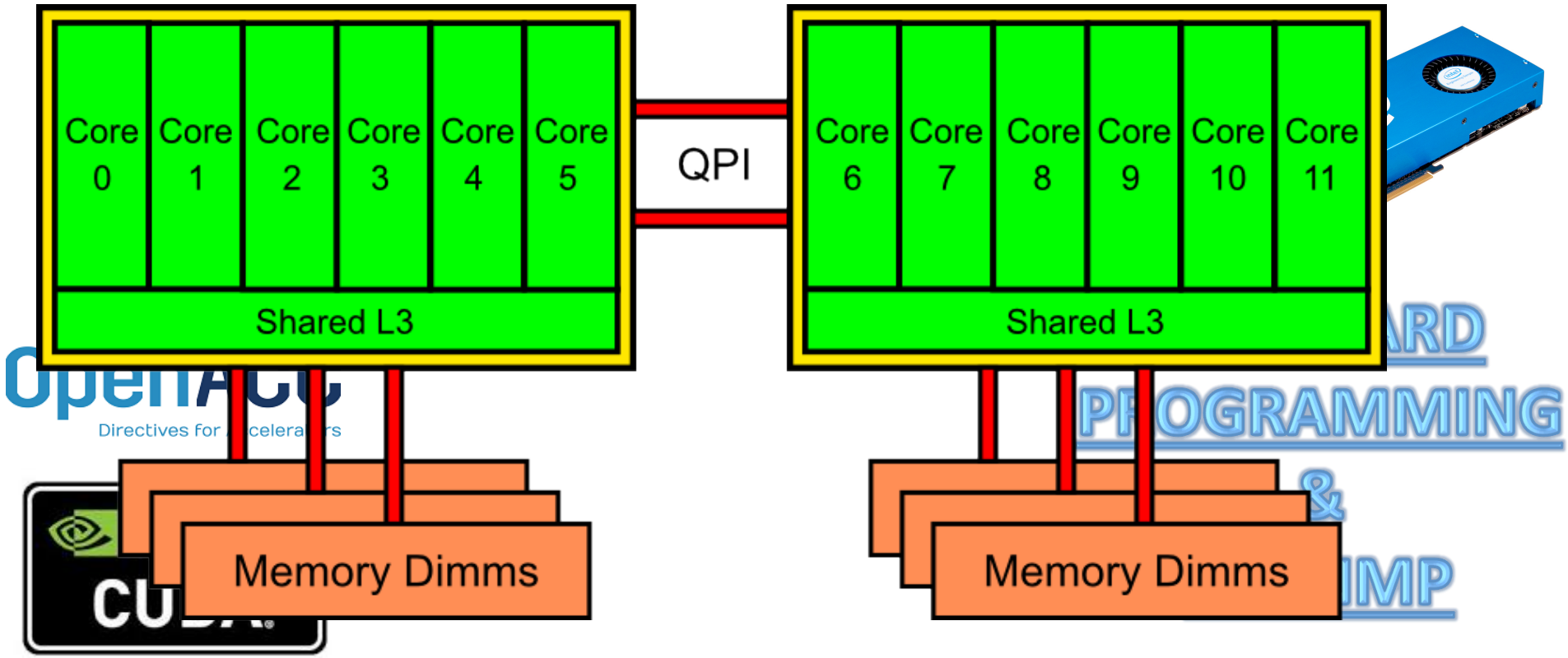



NETWORK
Memory Dimms
Shared L3
QPI

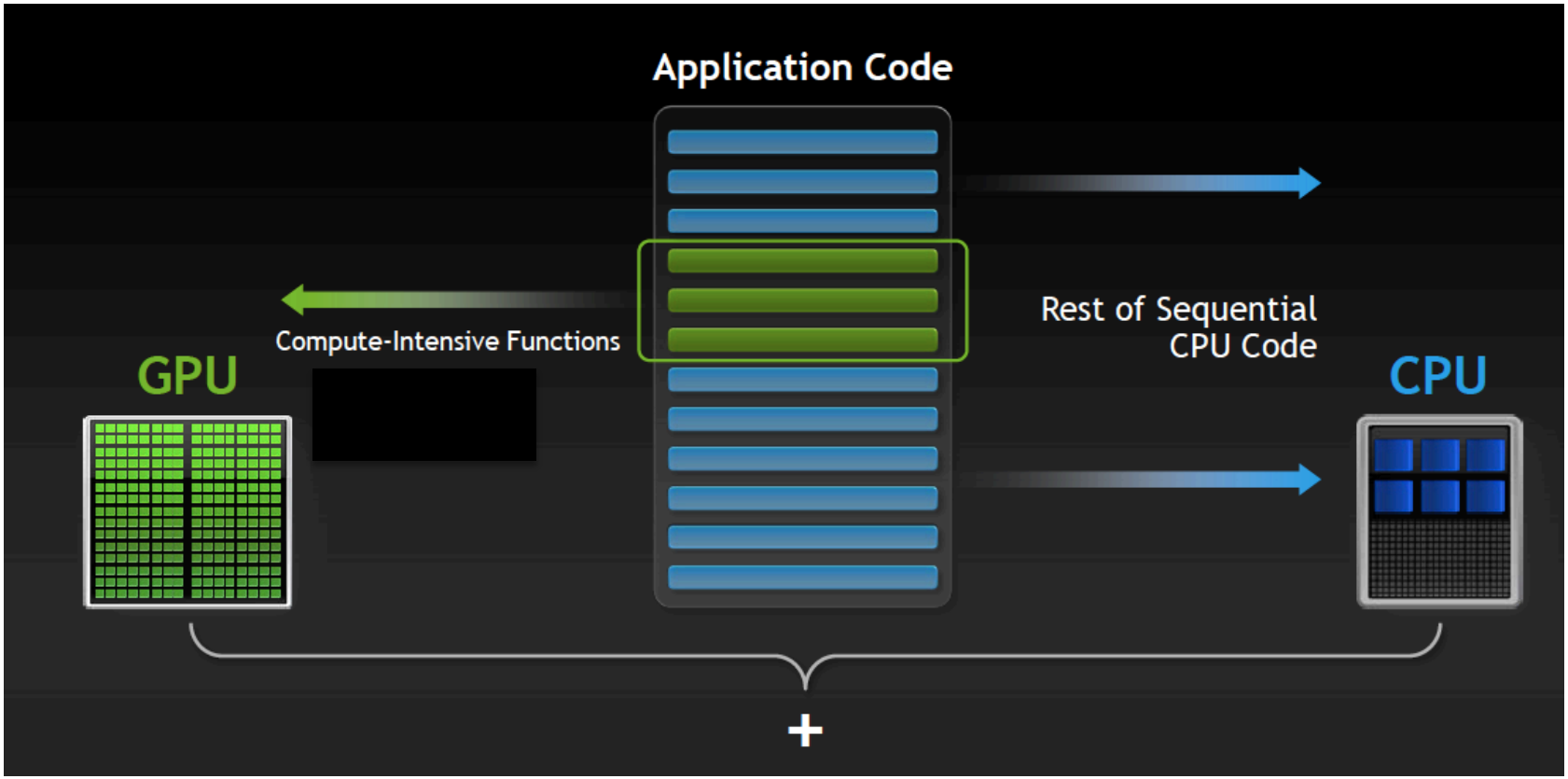
Threading and Vectorization

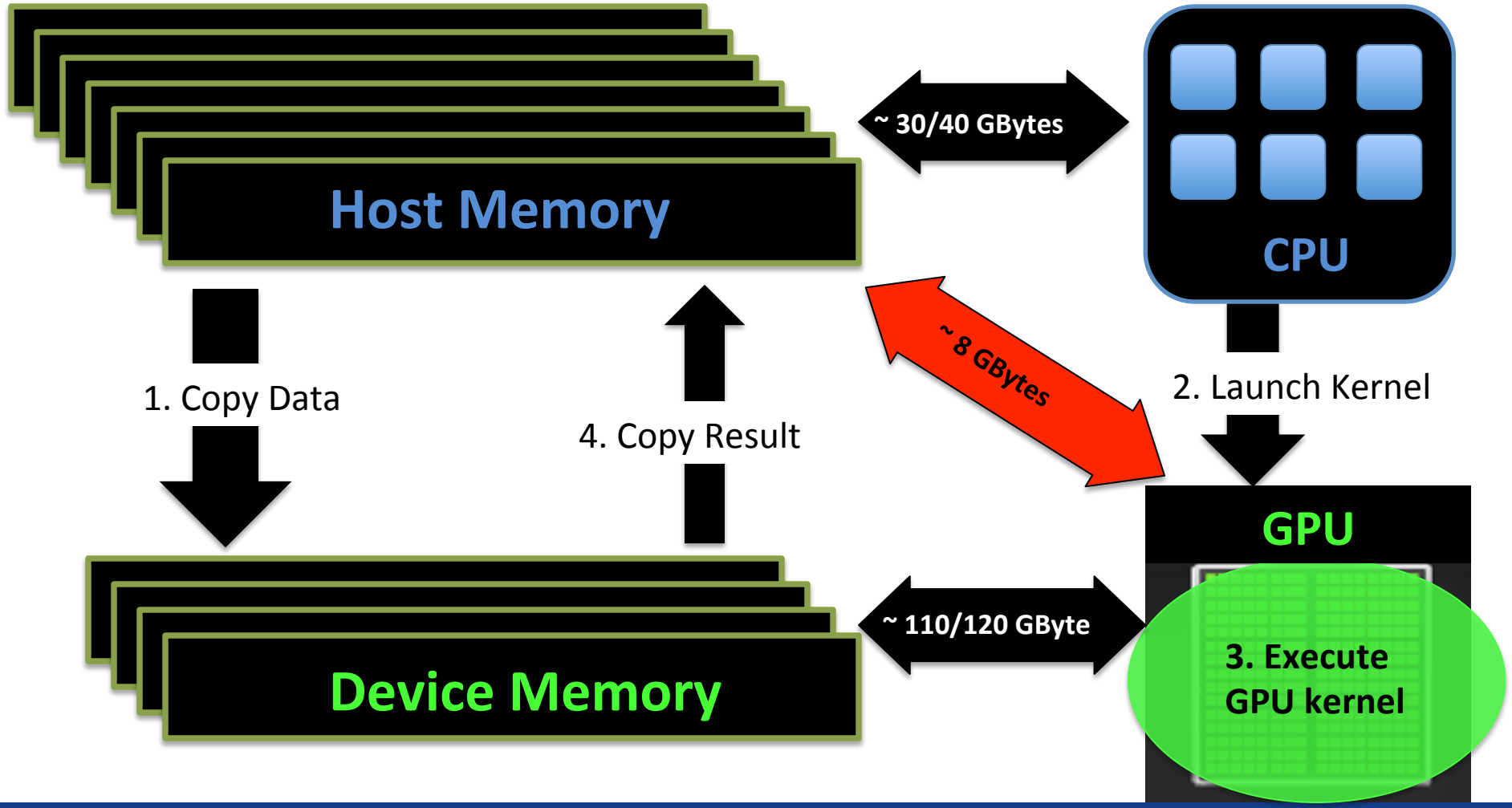


Multiple Socket CPUs + Acceleretors



The General Concept of Accelerated Computing




Host Memory
~ 30/40 GBytes
CPU
~ 8 GBytes
1. Copy Data
4. Copy Result
2. Launch Kernel
GPU
Device Memory
~ 110/120 GByte
3. Execute GPU kernel

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

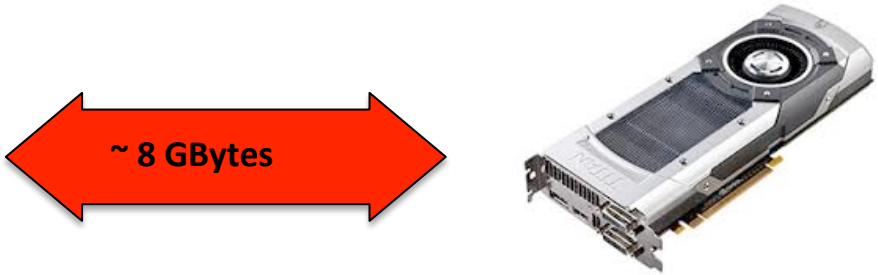
Why Does GPU Not Accelerate Computing?

- 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!!

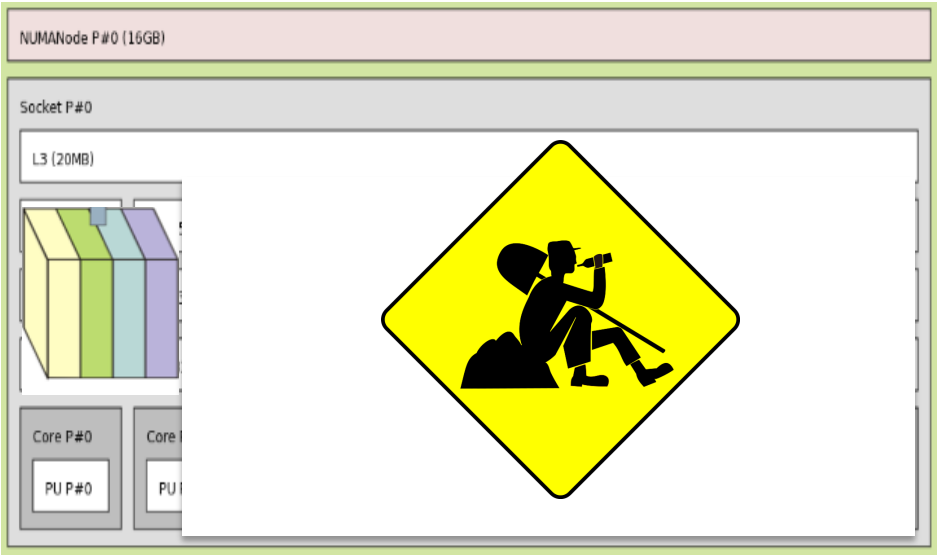
Higher aggregate computational power

- 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

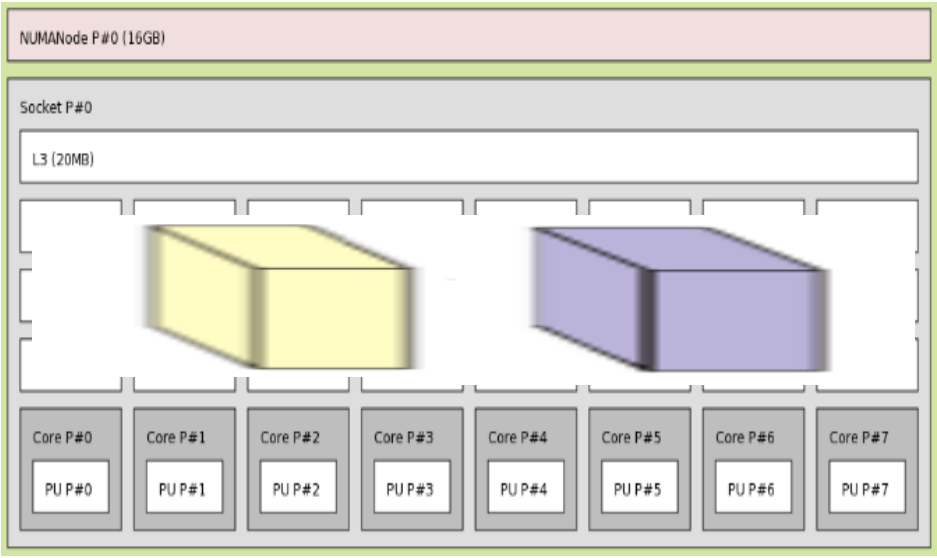



```
mpirun -np 8 pw-gpu.x -inp input file
```



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, workfows

MPI: Domain partition

OpenMP: Node Level shared mem

CUDA/OpenCL/OpenAcc: floating point accelerators

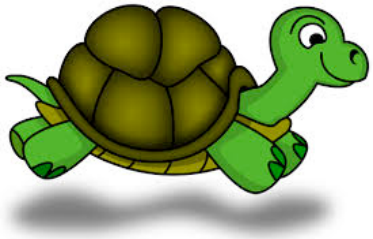
Summary

- 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

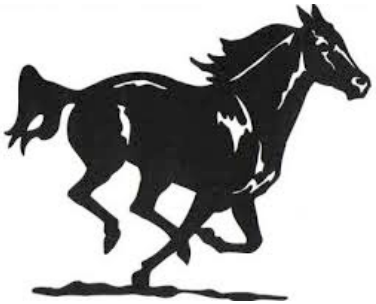
- Moore's law continues, but leads to multi-core, larger caches and higher integration => Performance increase now mostly through better algorithms, optimization, vectorization, and parallelization
- Bottleneck has transitioned from CPU speed to memory access and efficient data structures
- Write cleaned 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)

netlib



Freely available
Open Source Optimized

FFTW

MAGMA

ATLAS

OpenBLAS

PLASMA

Third-Party Highly-Optimized

AMD Core Math Library ACML

Intel Math Kernel Library

nag Results Matter. Trust NAG.

IBM ESSL

Thanks for your attention!!