The Abdus Salam
International Centre
for Theoretical Physics

Thinking in Parallel

Ivan Girotto – igirotto@ictp.it

Information & Communication Technology Section (ICTS)

International Centre for Theoretical Physics (ICTP)

Announcement

October 16, 2012 – 11AM – 12:15 Location: Leonardo Da Vinci Building, Euler Lecture Hall (Speaker: Ivan Girotto)
Subject - speed-up, efficiency and scalability. Understanding how to perform efficient simulations on parallel architecture.
Analysis based on a practical example: PWscf code from the Quantum ESPRESSO package.

--

October 23, 2012 – 11AM – 12:15 Location: Leonardo Da Vinci Building, Euler Lecture Hall (Speaker: Axel Kohlmeyer)

Subject - Introduction to HPC, why do we care and what is it about.
Overview of the main concepts are behind the fancy label "HPC"

--

October 30, 2012 – 11AM – 12:15 Location: Leonardo Da Vinci Building, Euler Lecture Hall (Speaker: David Grellscheid)
Subject - HPC in particle physics: computational tasks in High Energy Physics.
Overview of current strategies, and the features and problems of the LHC Computing Grid.

--

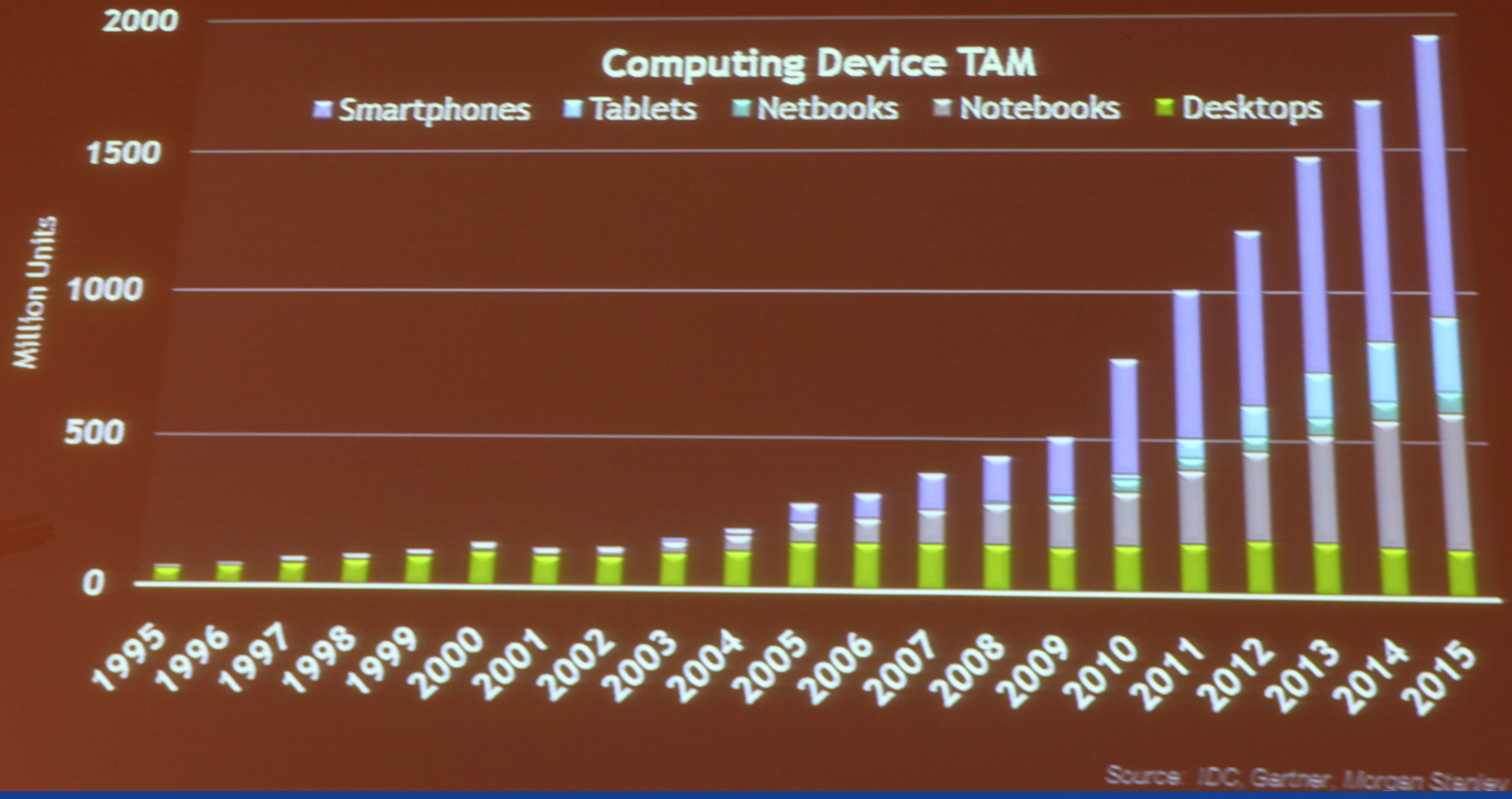
November 6, 2012 – 11AM – 12:15 Location: Leonardo Da Vinci Building, Euler Lecture Hall (Speaker: Axel Kohlmeyer)

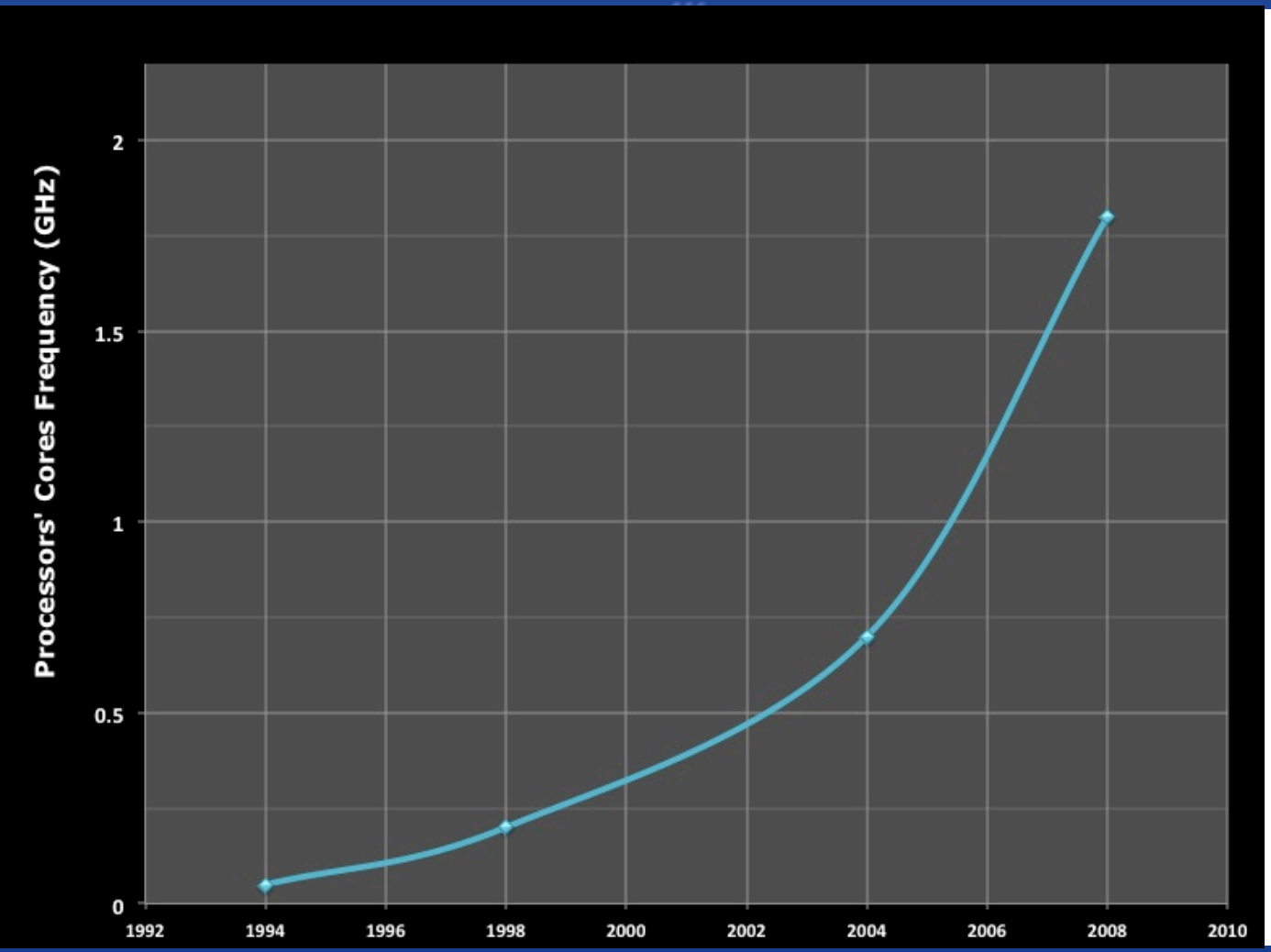
Subject: Axel will go through the most interesting experiences of his long career showing how HPC have had a significant impact for making real science.

Outline

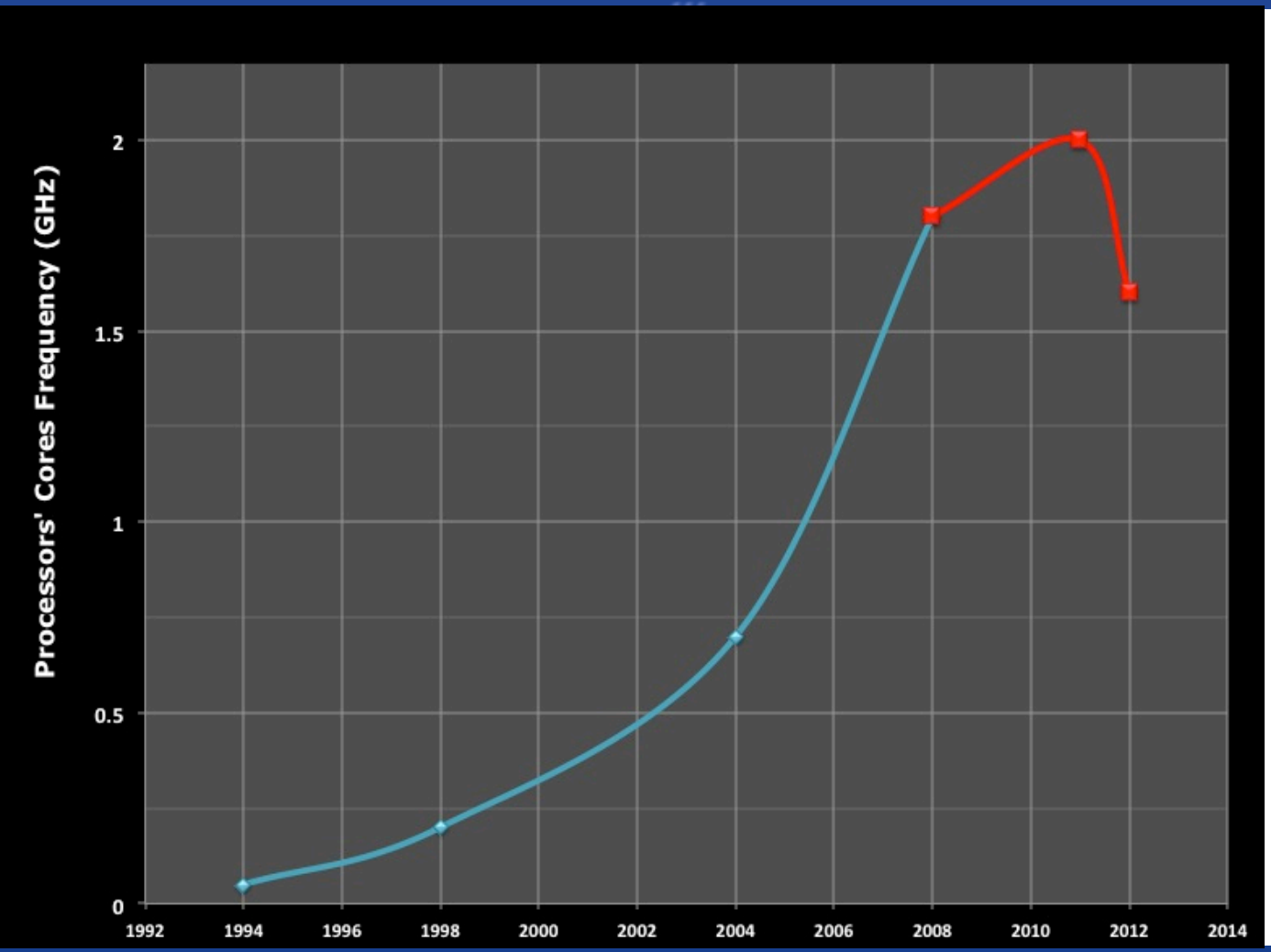
- Context: the multi-cores revolution
- Principles of parallelism
- Examples
- Conclusions

The Opportunity

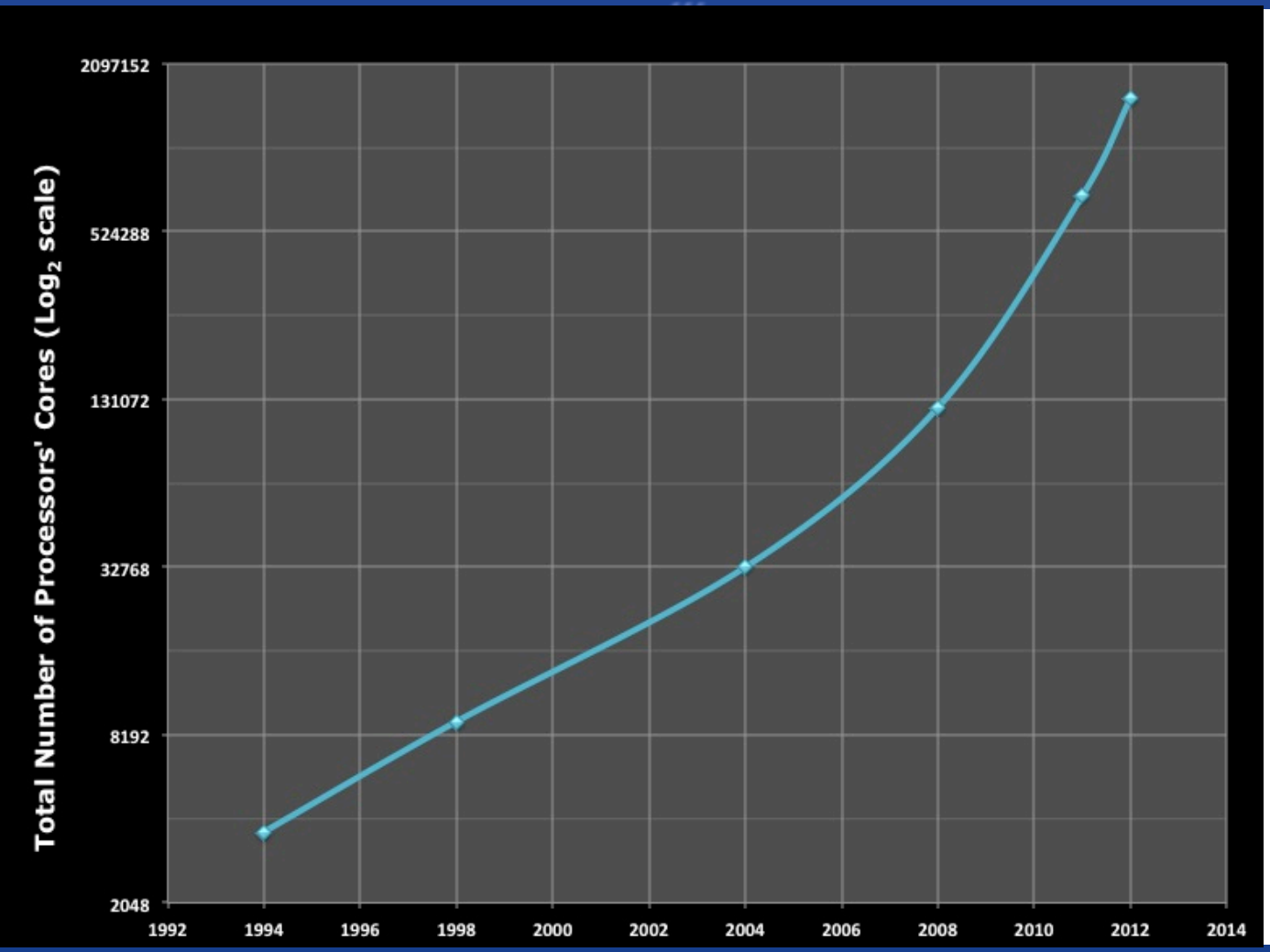




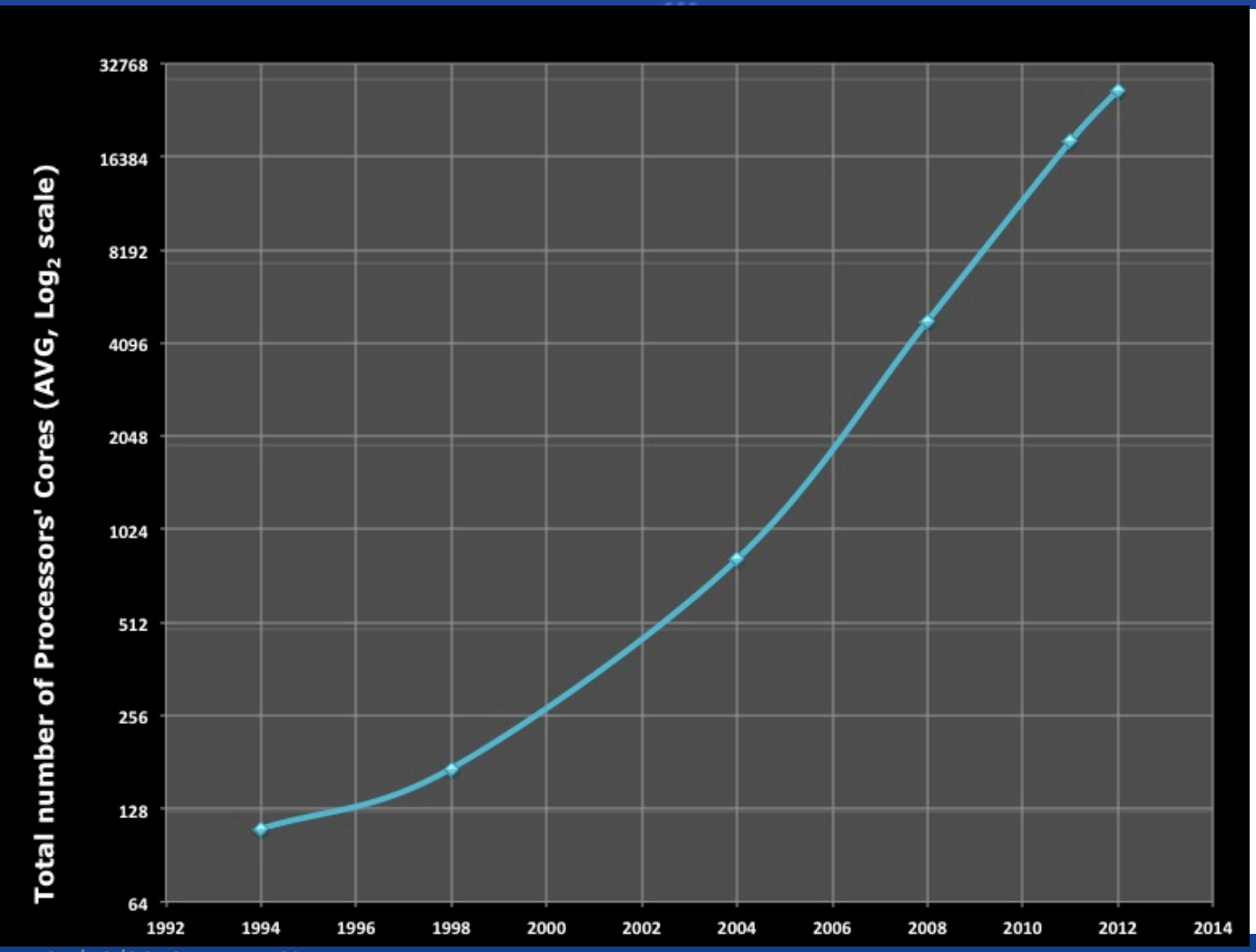
Processors' cores clock frequency to build the most powerful supercomputer in the world from 1994 to 2008. Data from www.top500.org



Processors' cores clock frequency to build the most powerful supercomputer in the world over the last 15 years. Data from www.top500.org



Total number of processors' cores to build the most powerful supercomputer in the world over the last 15 years. Data from www.top500.org



Average of the total number of processors' cores to build the 500 most powerful supercomputers in the world over the last 15 years. Data from www.top500.org

Consequences

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

Design of Parallel Algorithm /1

- A serial algorithm is a sequence of basic steps for solving a given problem using a single serial computer
- Similarly, a parallel algorithm is a set of instruction that describe how to solve a given problem using multiple (>=1) parallel processors
- The parallelism add the dimension of concurrency. Designer must define a set of steps that can be executed simultaneously!!!

Design of Parallel Algorithm /2

- Identify portions of the work that can be performed concurrently
- Mapping the concurrent pieces of work onto multiple processes running in parallel
- Distributing the input, output and intermediate data associated within the program
- Managing accesses to data shared by multiple processors
- Synchronizing the processors at various stages of the parallel program execution

Type of Parallelism

- **Functional (or task) parallelism**: different people are performing different task at the same time

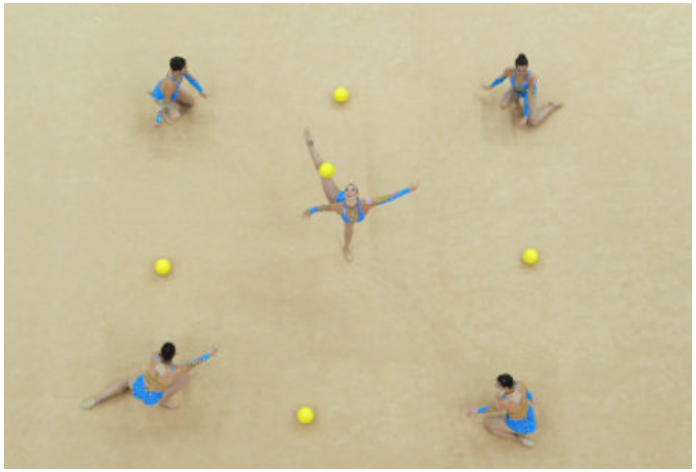
- **Data Parallelism**: different people are performing the same task, but on different equivalent and independent objects

Task/Process Mapping

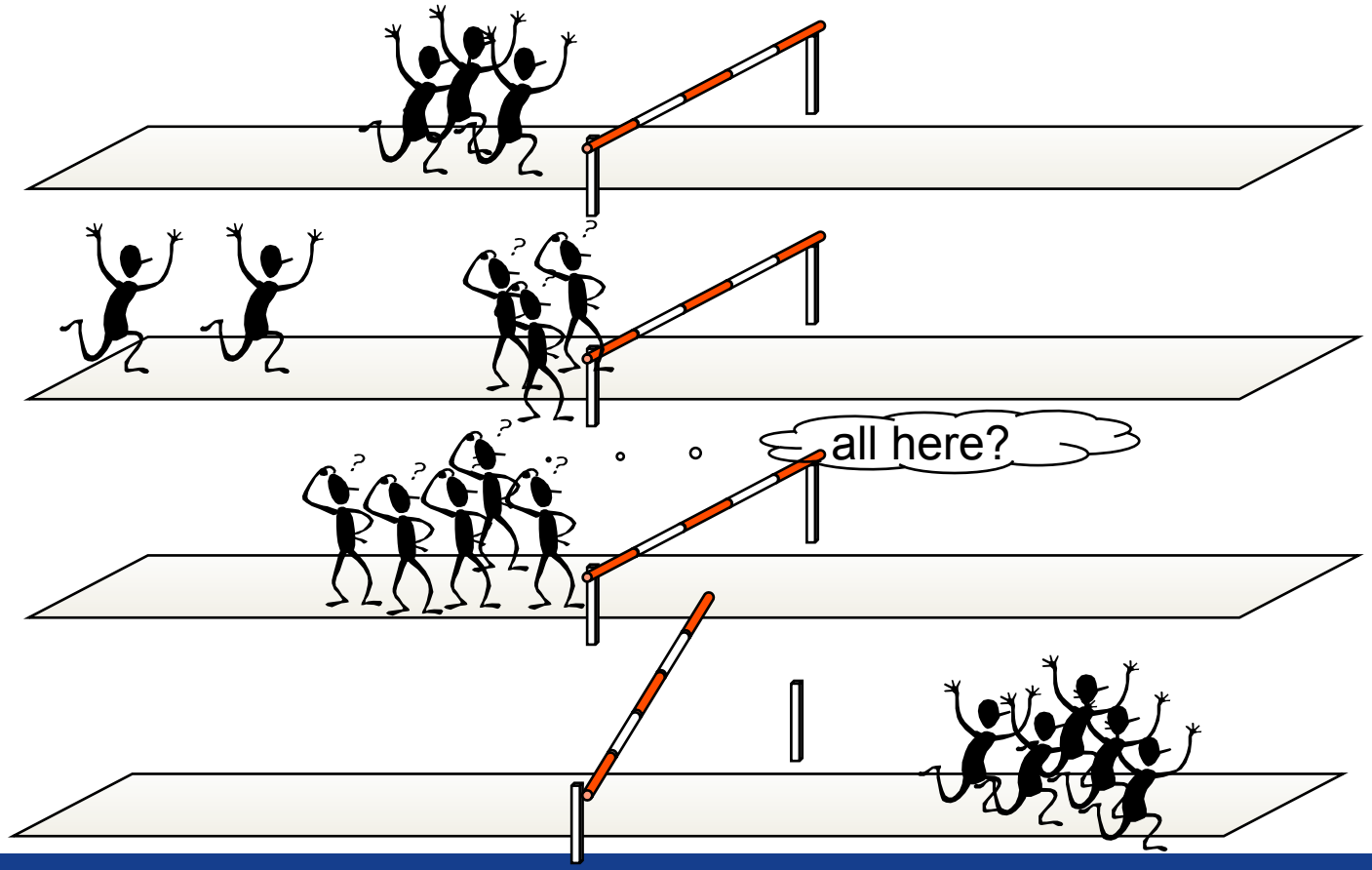
- The tasks, into which a problem is decomposed, are performed on physical processors
- The process is the computing agent that performs given tasks within a finite amount of time
- The mapping is the relation (N:N) by which tasks are assigned to processes for execution
- Mapping is one of the needed keys to find a decent load balancing. Mapping techniques are mainly:
  - Static Mapping: is defined prior to the execution (i.e., task distribution based on domain decomposition).
  - Dynamic Mapping: the work is distributed during the execution (i.e., master-slave model)

Process Interactions /1

- The effective speed-up obtained by the parallelization depend by the amount of overhead we introduce making the algorithm parallel
- There are mainly two key sources of overhead:
  1. Time spent in inter-process interactions (**communication**)
  2. Time some process may spent being idle (**synchronization**)



What happens if the girls are not well trained?



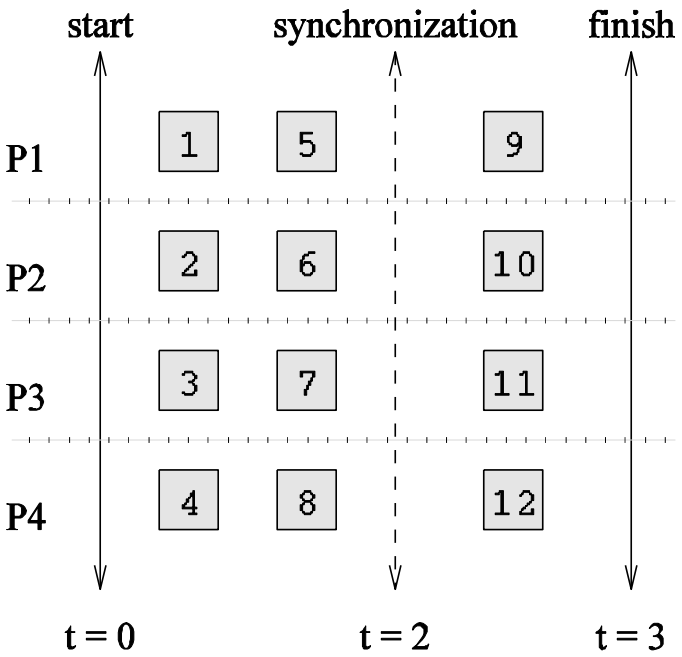
Process Interactions /2

Synchronization

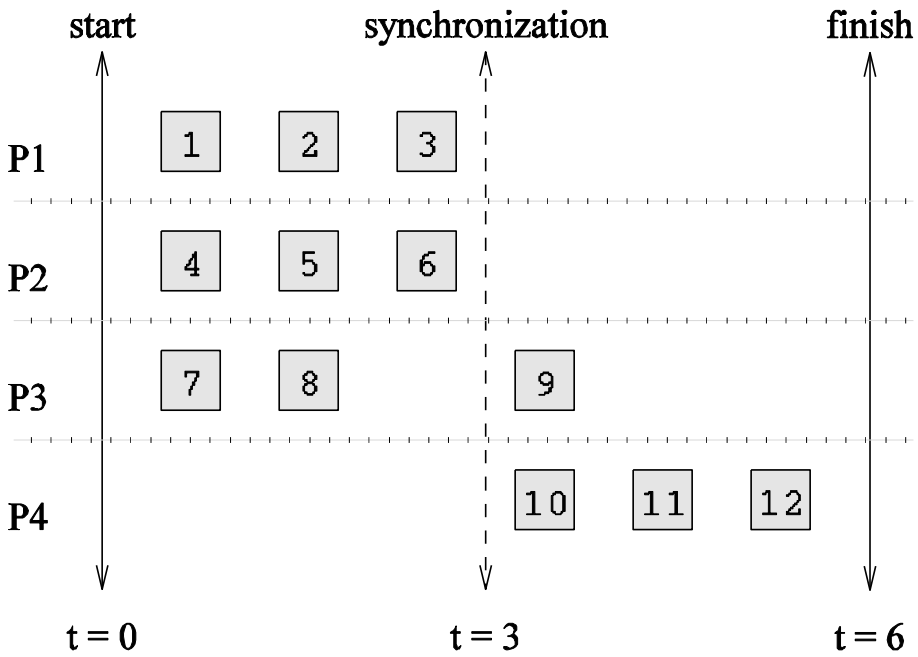
+

Communication

Mapping and Synchronization



(a)

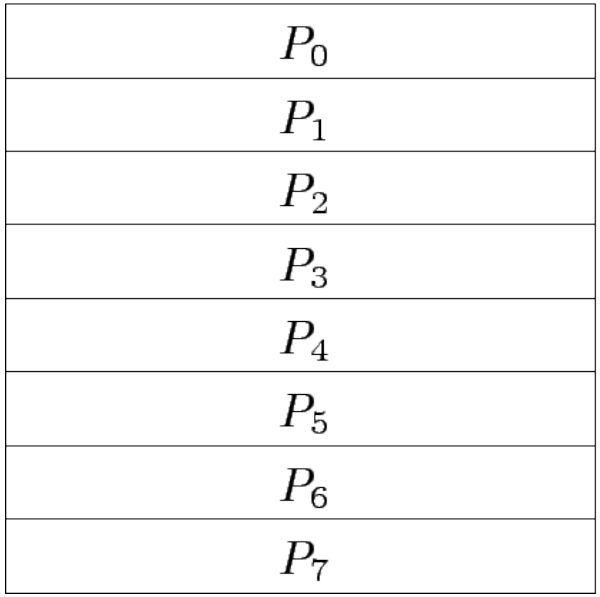


(b)

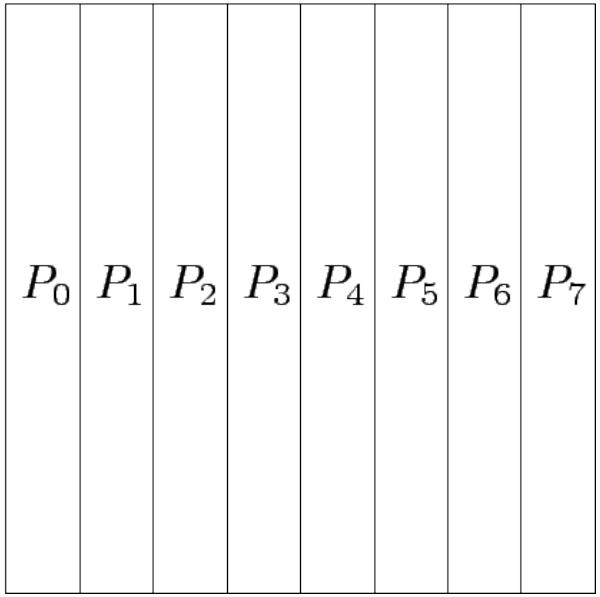
Static Data Partitioning

The simplest data decomposition schemes for dense matrices are 1-D block distribution schemes.

row-wise distribution

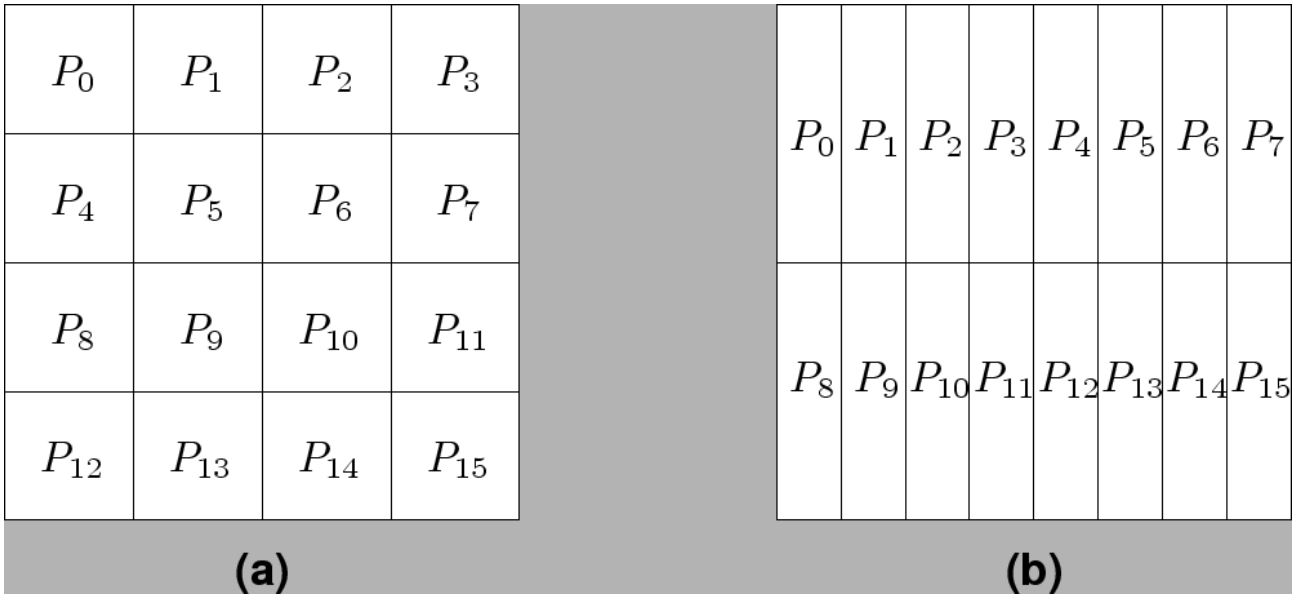


column-wise distribution



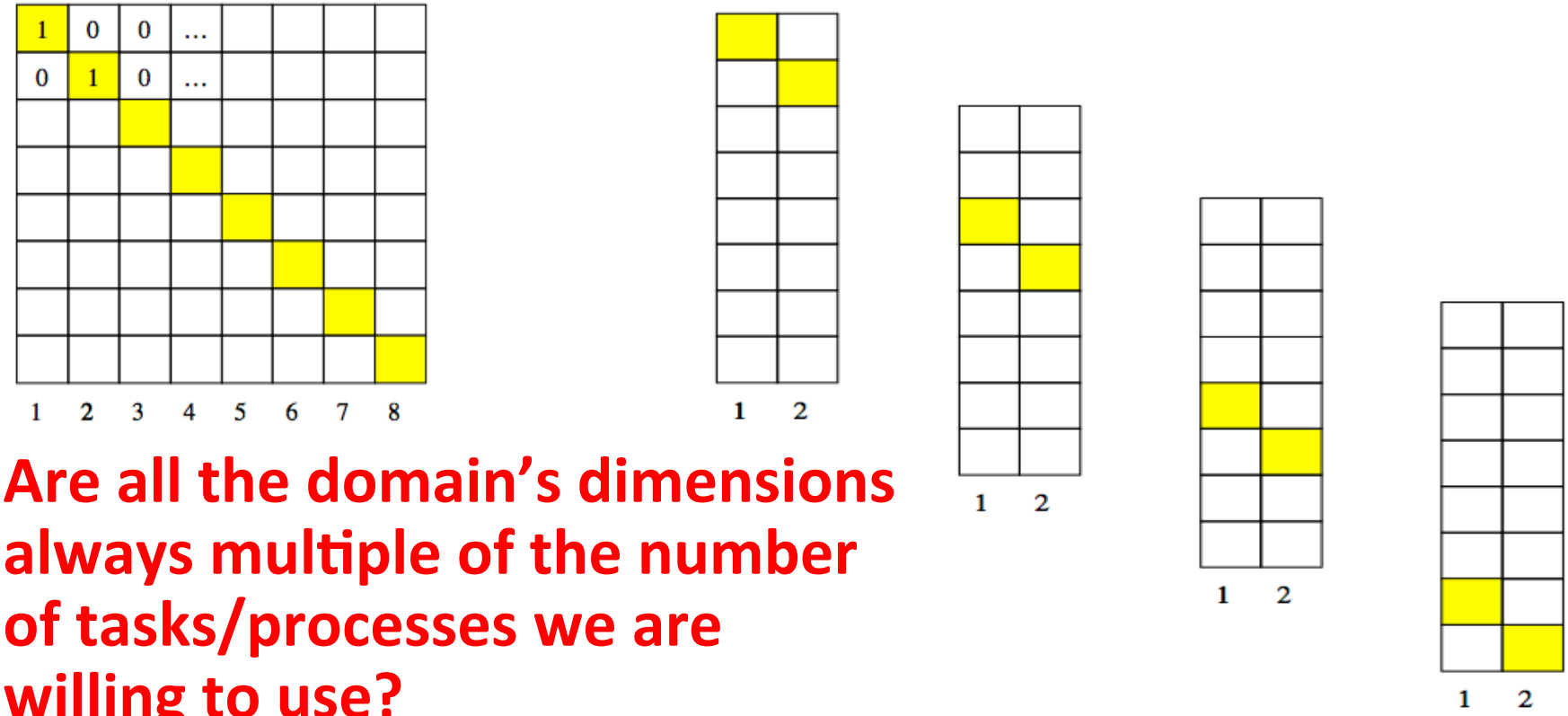
Block Array Distribution Schemes

Block distribution schemes can be generalized to higher dimensions as well.



Degree to which tasks/data can be subdivided is limit to concurrency and parallel execution!!

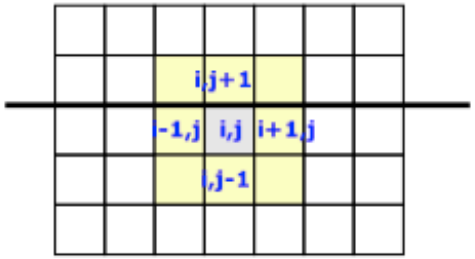
Collaterals to Domain Decomposition /1

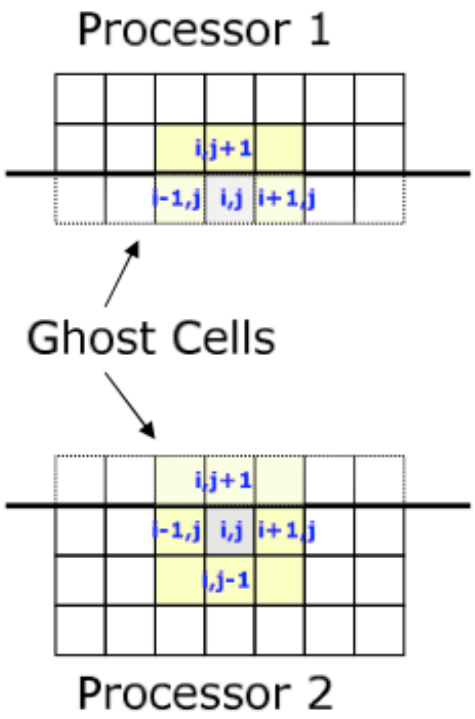


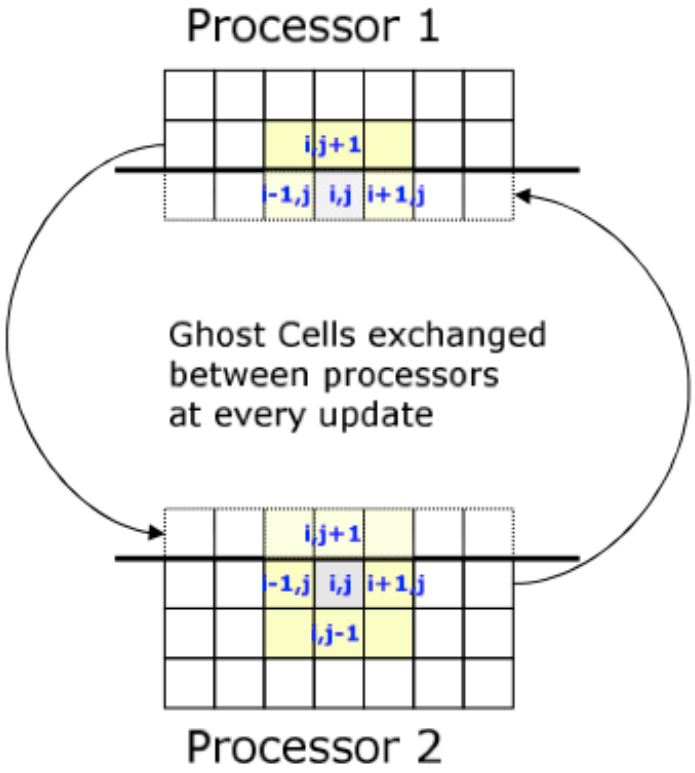
Are all the domain's dimensions always multiple of the number of tasks/processes we are willing to use?

Collaterals to Domain Decomposition /2

sub-domain boundaries



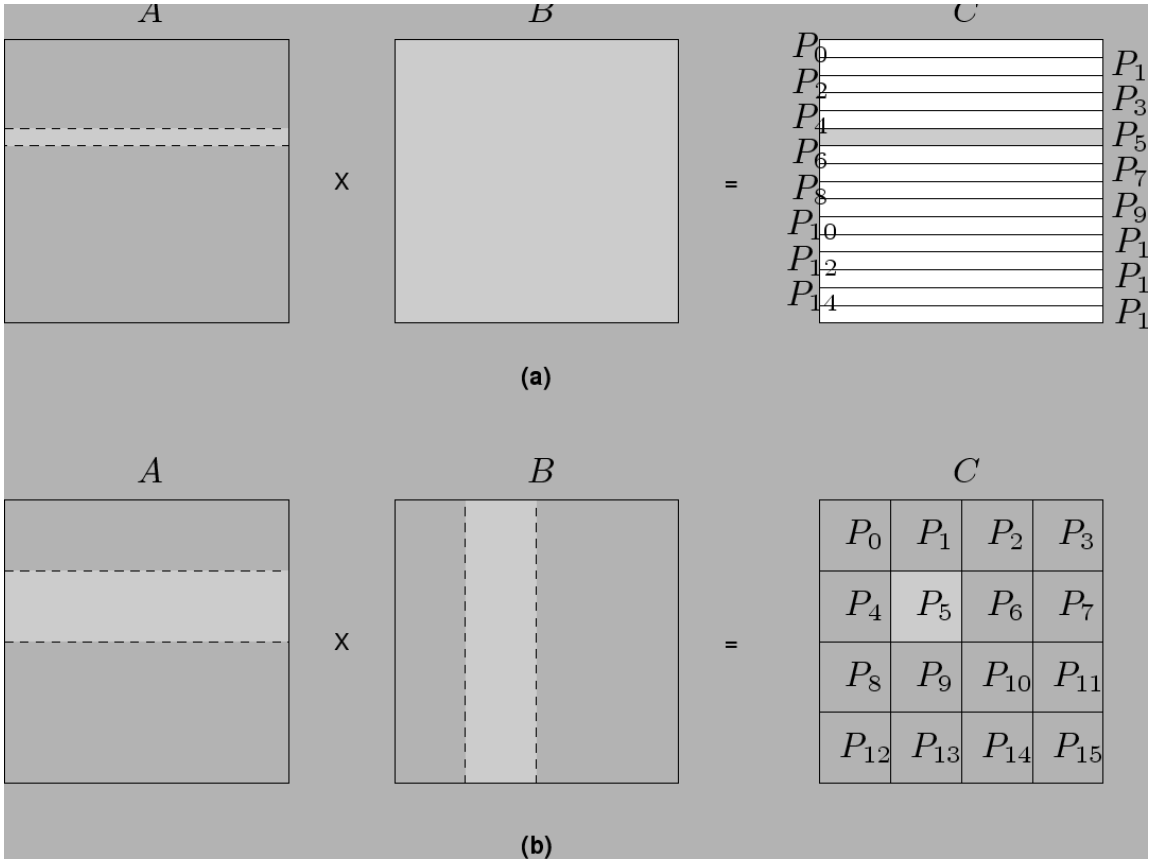




Block Array Distribution Schemes: GEMM

- For multiplying two dense matrices ***A*** and ***B***, we can partition the output matrix ***C*** using a block decomposition.
- For load balance, we give each task the same number of elements of ***C***. (Note that each element of ***C*** corresponds to a single dot product.)
- The choice of precise decomposition (1-D or 2-D) is determined by the associated communication overhead.
- In general, higher dimension decomposition allows the use of larger number of processes.

Data Sharing in Dense Matrix Multiplication



Granularity

- Granularity is determined by the decomposition level (number of task) on which we want divide the problem
- The degree to which task/data can be subdivided is limit to concurrency and parallel execution
- Parallelization has to become “topology aware”
  - coarse grain and fine grained parallelization has to be mapped to the topology to reduce memory and I/O contention

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

The Fast Fourier Transform (FFT) /1

Consider a sequence X = X[0],X[1],X[2],...,X[n−1] of length n. The 1D Discrete Fourier Transform of the sequence X is the sequence Y = Y [0],Y [1],Y [2],...,Y [n−1] where:

$$Y[i] = \sum_{k=0}^{n-1} X[k]\,\omega^{ki} \qquad 0 \leq i < n$$

with

$$\omega = e^{2\pi\sqrt{-1}/n}$$

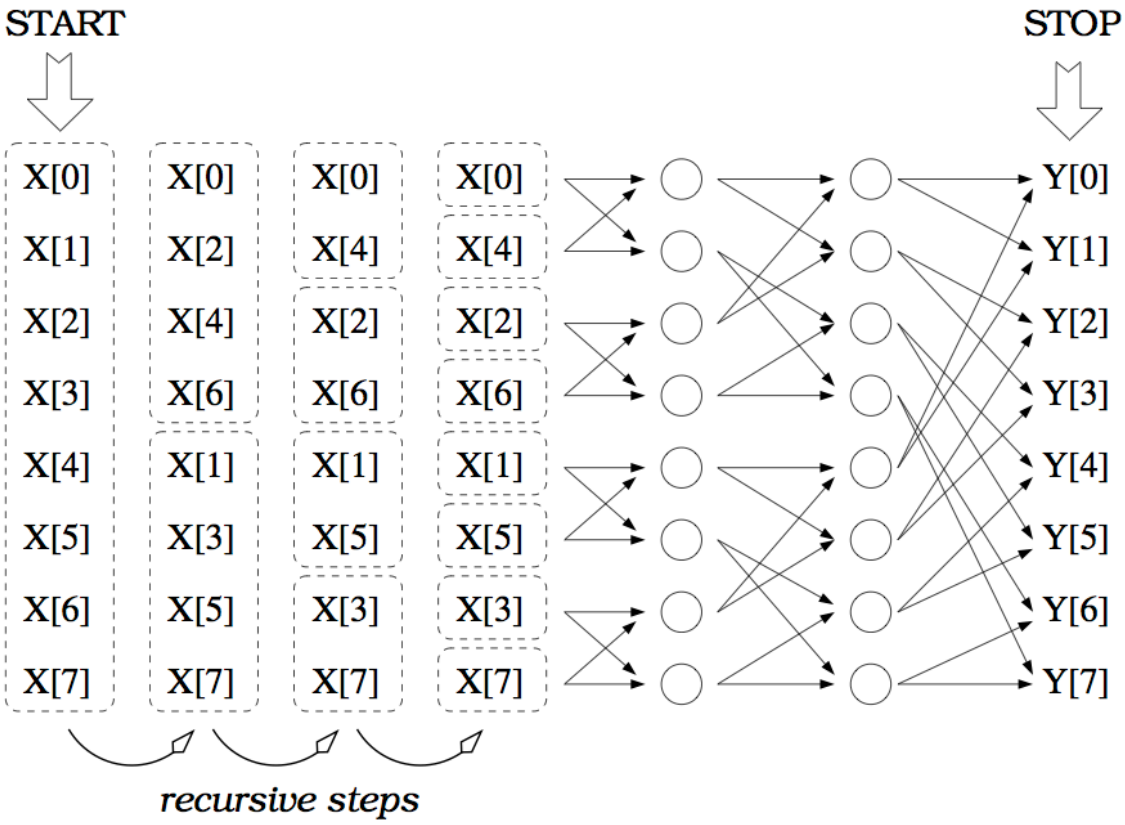
The Fast Fourier Transform (FFT) /2

$$Y[i] = \sum_{k=0}^{(n/2)-1} X[2k]\,\omega^{2ki} + \sum_{k=0}^{(n/2)-1} X[2k+1]\,\omega^{(2k+1)i}$$

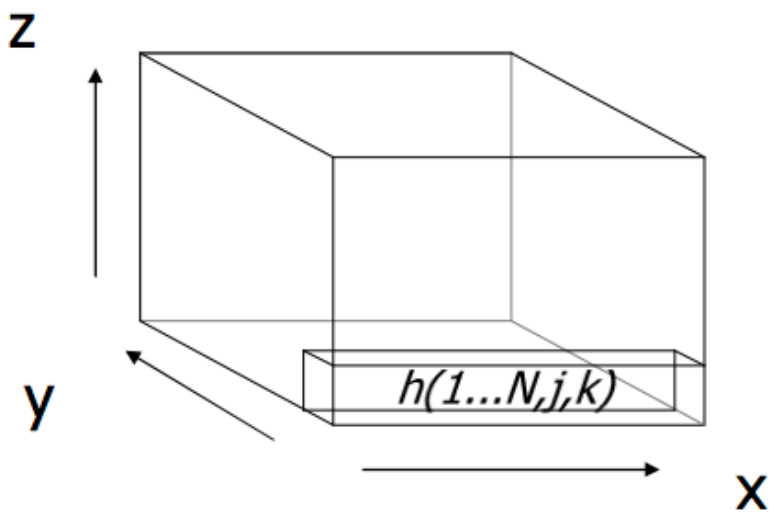
$$= \sum_{k=0}^{(n/2)-1} X[2k]\,e^{2\pi\sqrt{-1}ki/(n/2)} + \omega^{i} \sum_{k=0}^{(n/2)-1} X[2k+1]\,e^{2\pi\sqrt{-1}ki/(n/2)}$$

… but if *n* is power of 2 (*2-radix*) each (n/2)-DFT can be divided recursively until we have the single component.

The Fast Fourier Transform (FFT) /3



Multidimensional FFT



$$f(x,y,z) = \frac{1}{N_z N_y N_x} \sum_{z=0}^{N_z-1} \Big(\sum_{y=0}^{N_y-1} \Big(\underbrace{\sum_{x=0}^{N_x-1} F(u,v,w)\, e^{-2\pi i \frac{xu}{N_x}}}_{\text{DFT long x-dimension}} \Big) e^{-2\pi i \frac{yv}{N_y}} \Big) e^{-2\pi i \frac{zw}{N_z}}$$

DFT long y-dimension

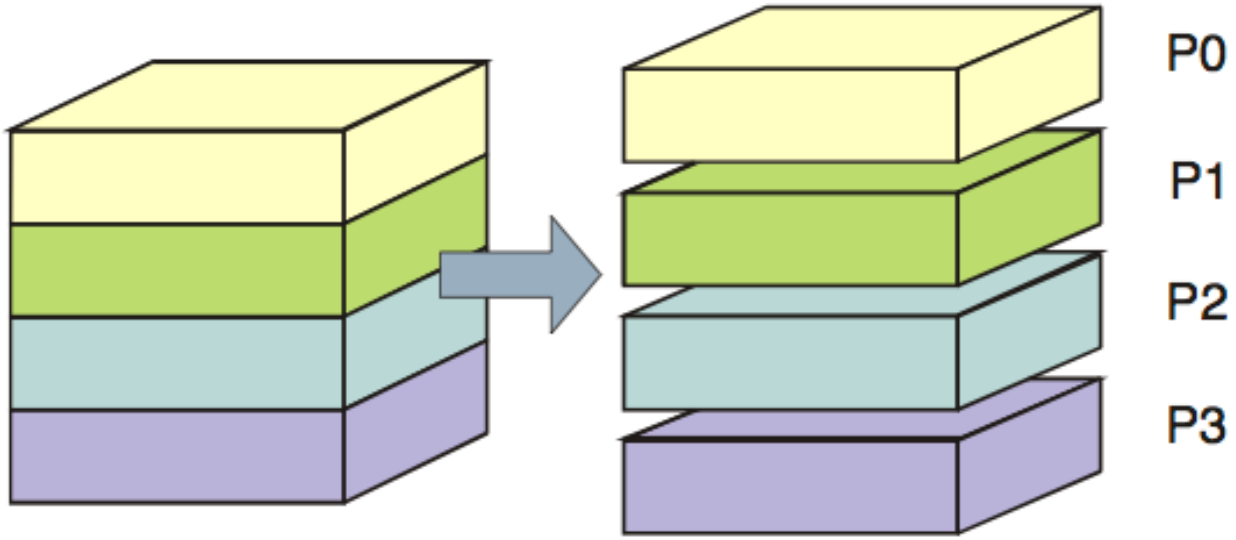
DFT long z-dimension

1) For any value of ***j*** and ***k*** transform the column (1…N, j, k)

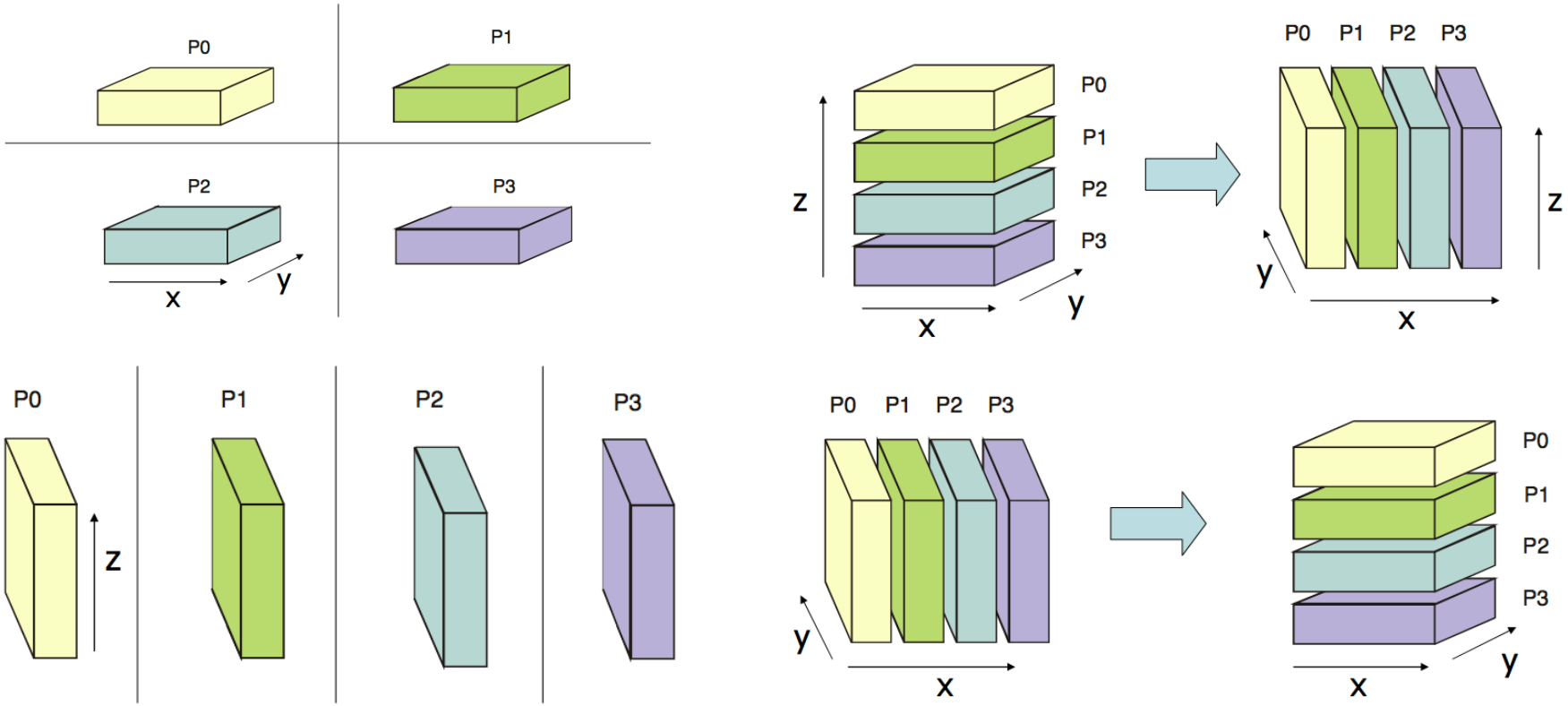
2) For any value of ***i*** and ***k*** transform the column (i, 1…N, k)

3) For any value of ***i*** and ***j*** transform the column (i, j, 1…N)

Parallel 3DFFT / 1



Parallel 3DFFT / 2



http://www.fftw.org/

FFTW

Download Mailing List Benchmark Features Documentation FAQ Links Feedback

Introduction

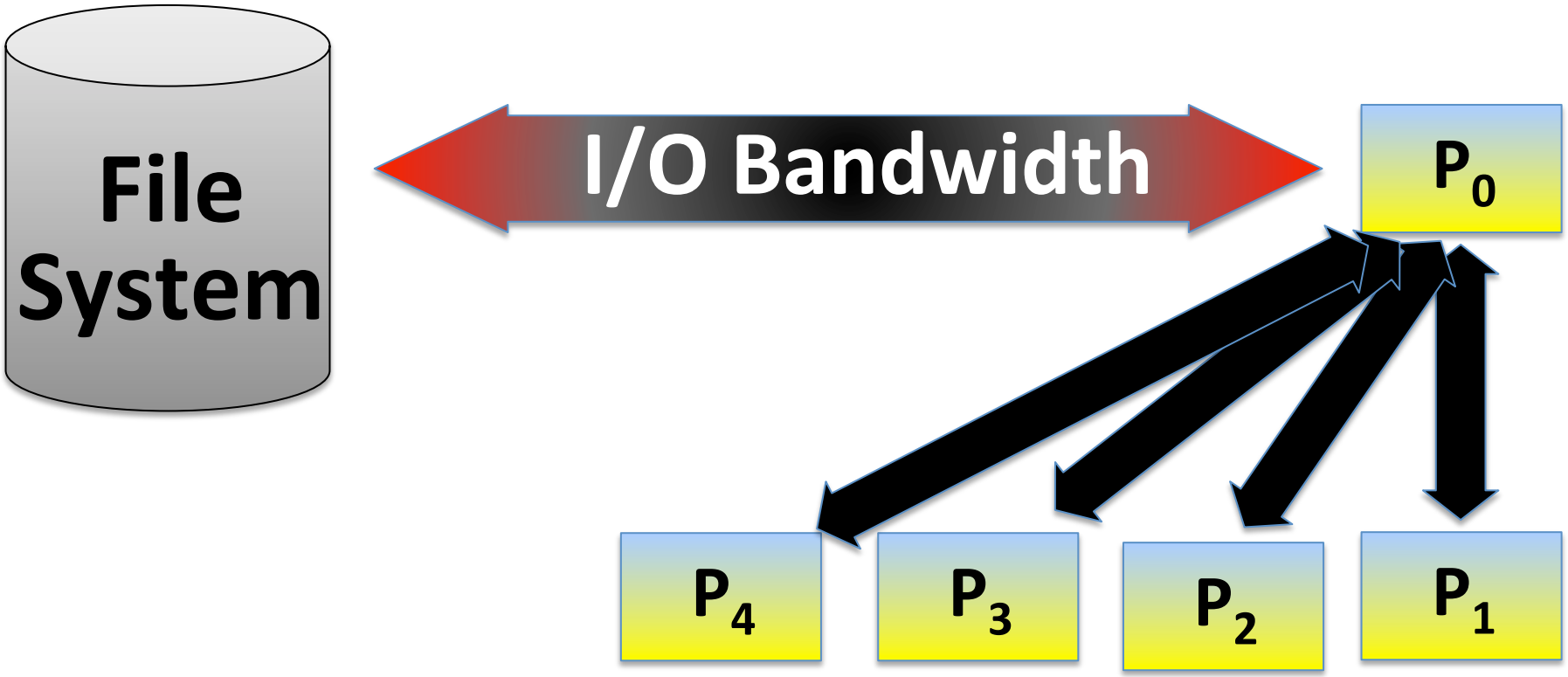
FFTW is a C subroutine library for computing the discrete Fourier transform (DFT) in one or more dimensions, of arbitrary input size, and of both real and complex data (as well as of even/odd data, i.e. the discrete cosine/sine transforms or DCT/DST). We believe that FFTW, which is free software, should become the FFT library of choice for most applications.

The latest official release of FFTW is version **3.3.2**, available from our download page. Version 3.3 introduces support for the AVX x86 extensions, a distributed-memory implementation on top of MPI, and a Fortran 2003 API. Version 3.3.1 introduces support for the ARM Neon extensions. See the release notes for more information.

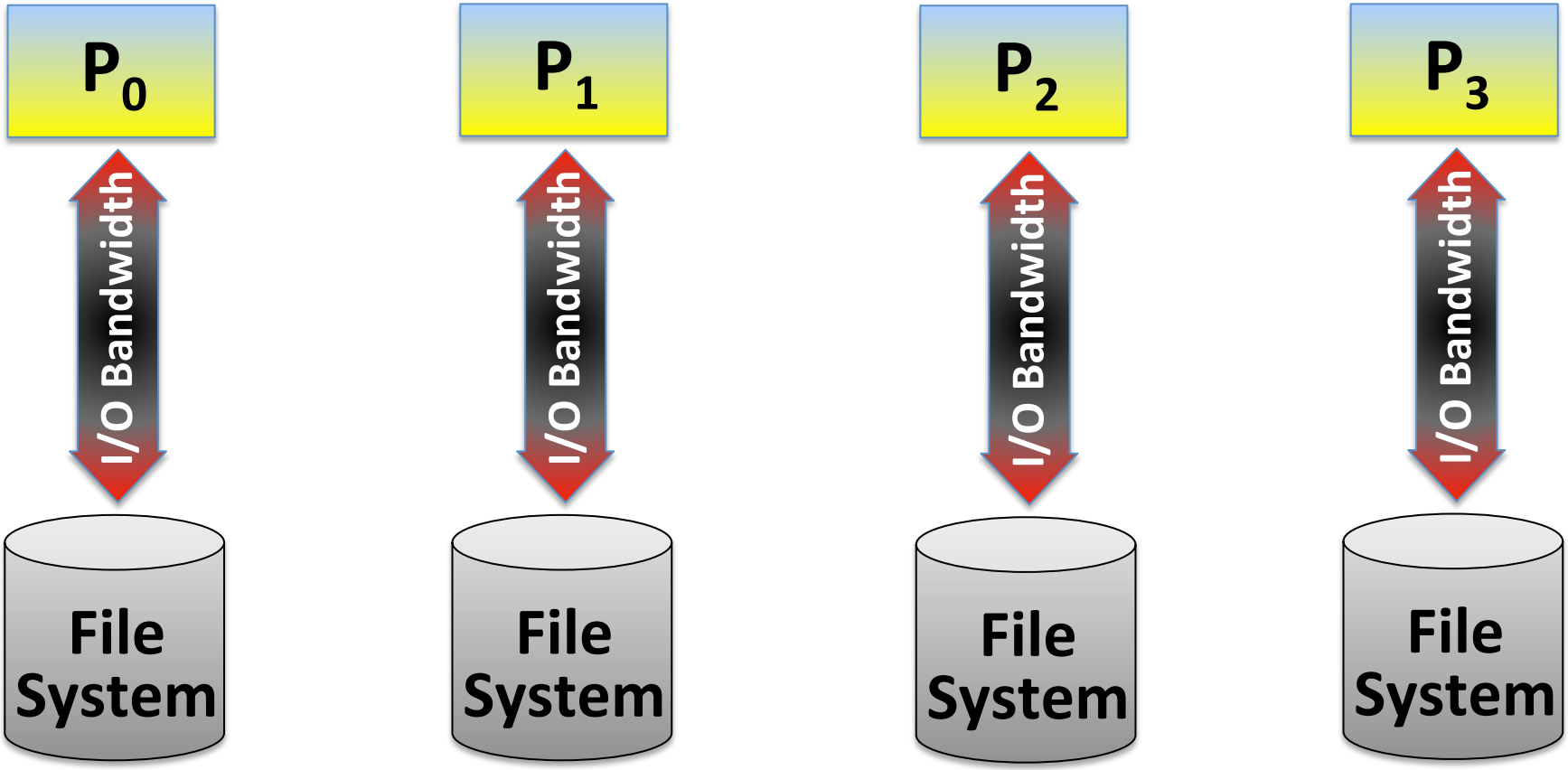
The FFTW package was developed at MIT by Matteo Frigo and Steven G. Johnson.

Our benchmarks, performed on on a variety of platforms, show that FFTW's performance is typically superior to that of other publicly available FFT software, and is even competitive with vendor-tuned codes. In contrast to vendor-tuned codes, however, FFTW's performance is *portable*: the same program will perform well on most architectures without modification. Hence the name, "FFTW," which stands for the somewhat whimsical title of "**Fastest Fourier Transform in the West**."

Parallel I/O

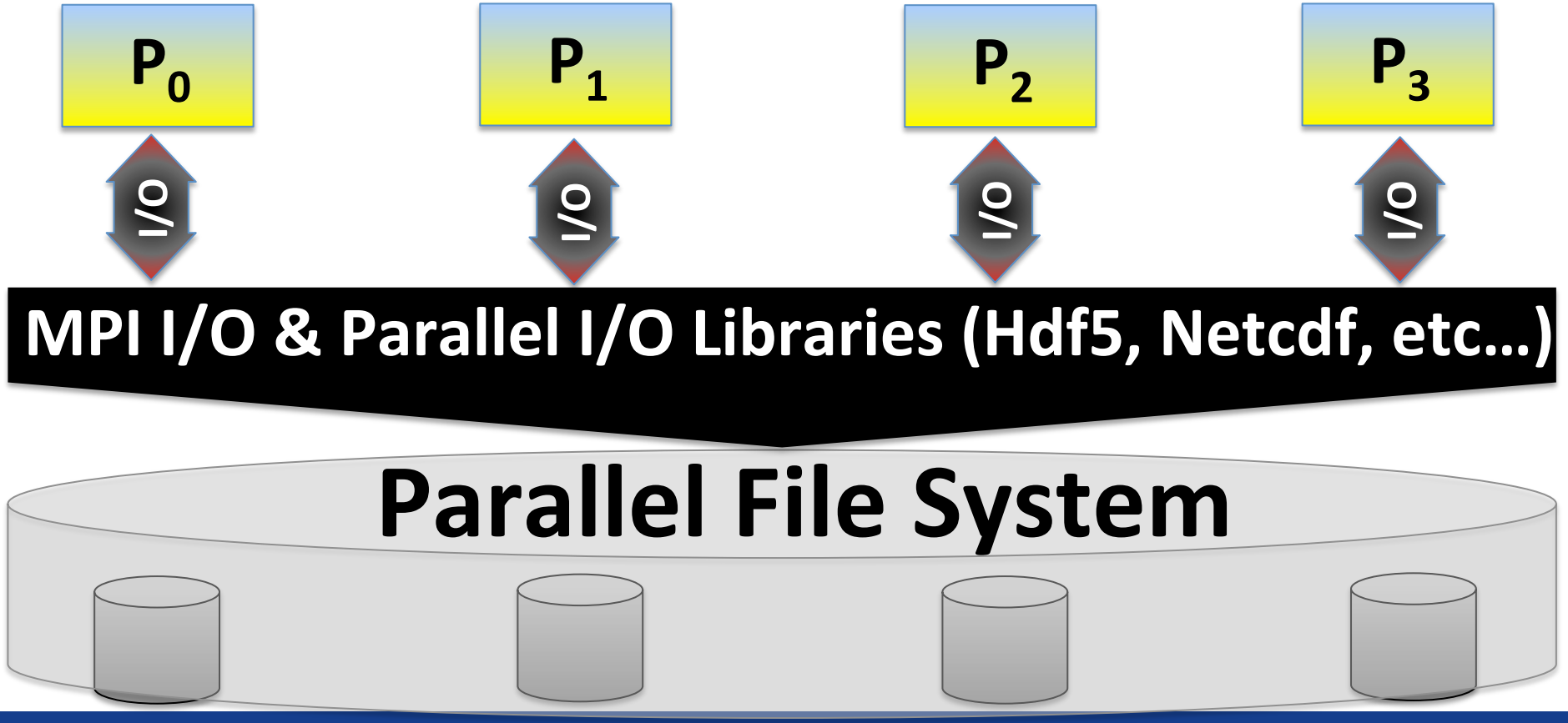


Parallel I/O


P0
P1
P2
P3
I/O Bandwidth
I/O Bandwidth
I/O Bandwidth
I/O Bandwidth
File System
File System
File System
File System

Serial I/O

Parallel I/O


P0
P1
P2
P3
I/O
I/O
I/O
I/O
MPI I/O & Parallel I/O Libraries (Hdf5, Netcdf, etc…)
Parallel File System

Conclusions

- The technology evolution/revolution do not allow to longer work around the parallelism
- Thinks at parallel algorithms as well as at parallel solutions
- Make your code modularized to enhance different levels of granularity and consequently to become more "platform adaptable"

Thanks for your attention!!

References

- Filippo Spiga, *Implementing and testing Mixed Parallel Programming Model into Quantum ESPRESSO*, Msc Thesis, April 9, 2004 - http://filippospiga.me

- A. Grama, A. Gupta, G. Karypis, and V. Kumar, *Introduction to Parallel Computing*, Pearson Education, 2003, ISBN: 0201648652, 9780201648652