The Abdus Salam International Centre for Theoretical Physics

Use of QE in HPC: trend in technology for HPC, basics of parallelism and performance features

Ivan Girotto – igirotto@ictp.it

Information & Communication Technology Section (ICTS)

International Centre for Theoretical Physics (ICTP)

Outline

- Trends of Computing Platforms in HPC
- Principles of Parallelism in SW Applications
- Applied High-Performance and Parallel Computing in the QE Distribution
- Performance Analysis
- Conclusions

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

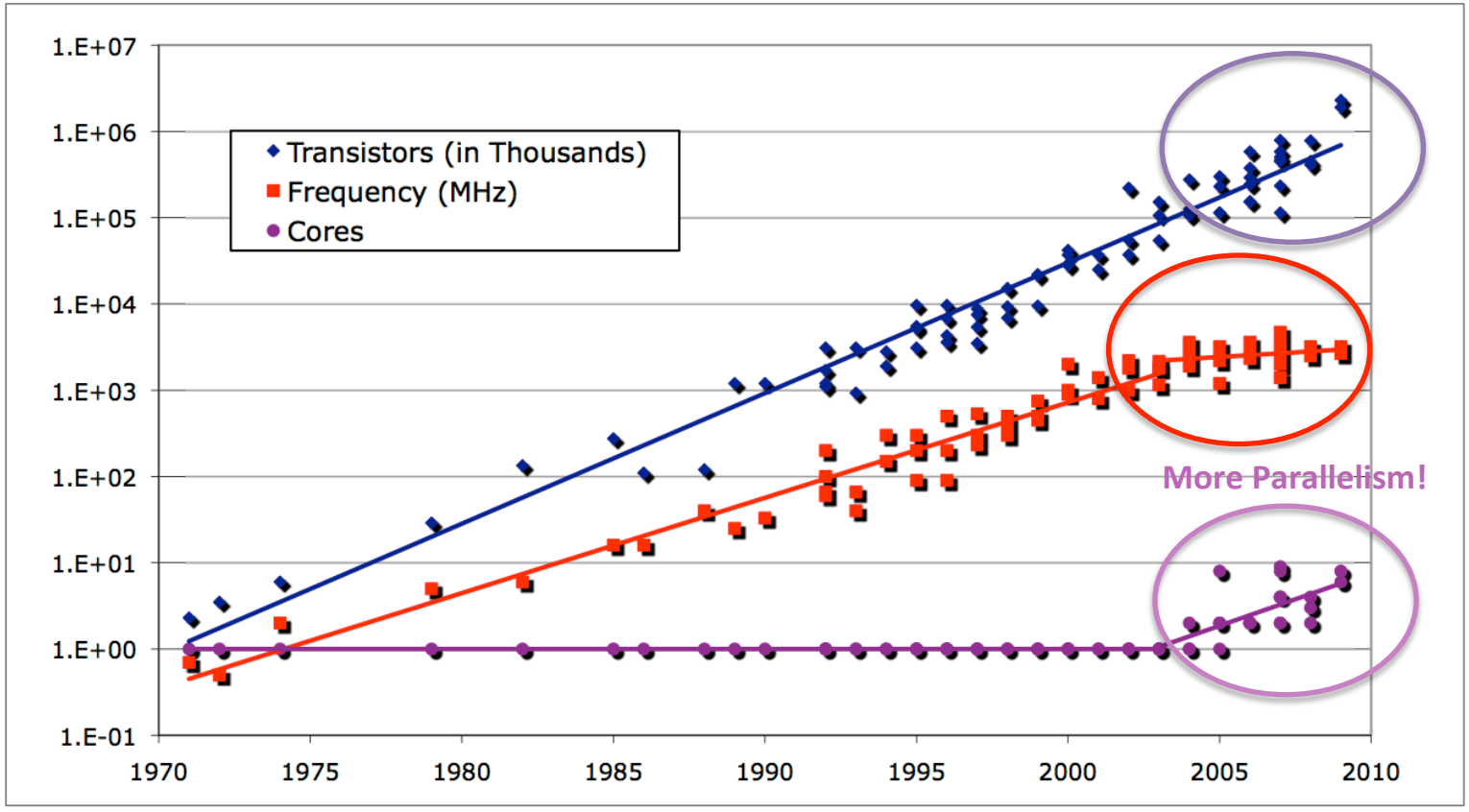
Why would HPC matter to you?

- 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

HPC Trend and Moore's Law



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

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

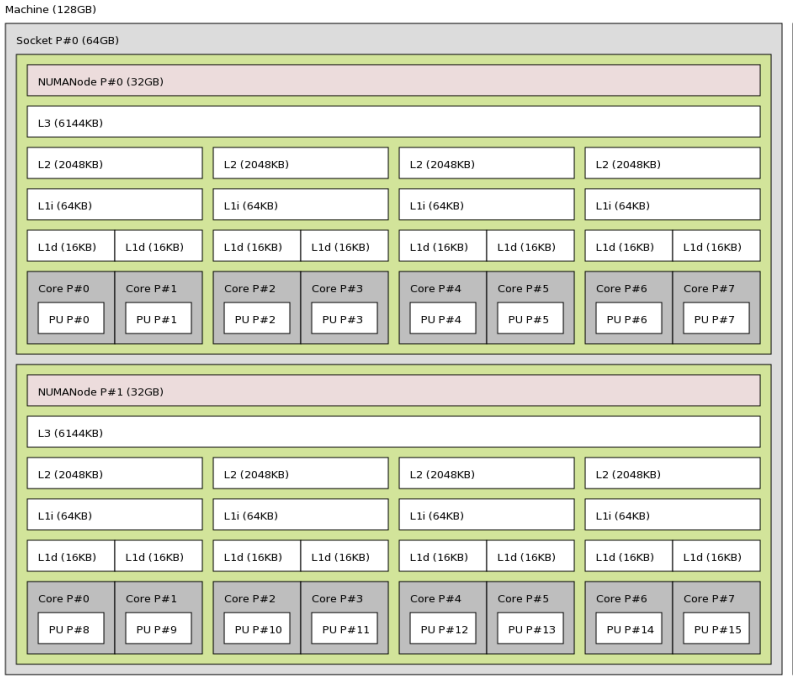
NVIDIA → GPU alone will not last long
ARM+GPU, Power+GPU

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

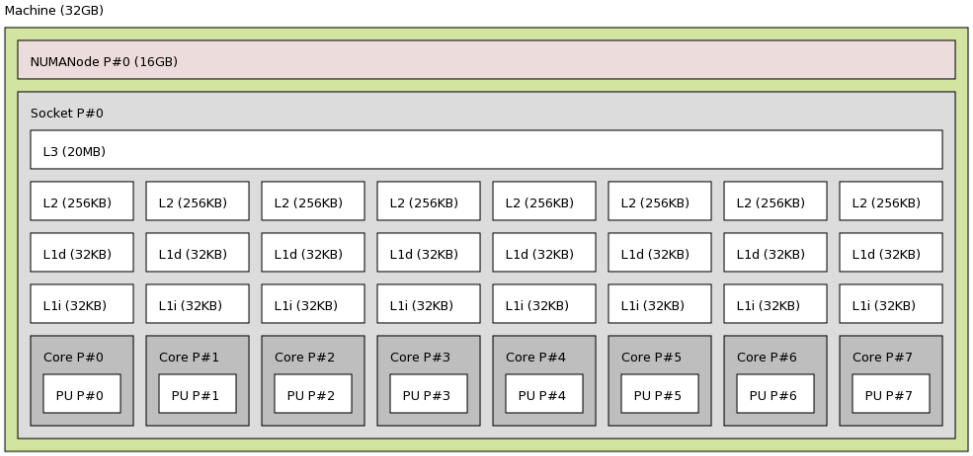
AMD → Console market
Still some chance for HPC

PARALLEL COMPUTER PLATFORMS

Modern CPUs Models



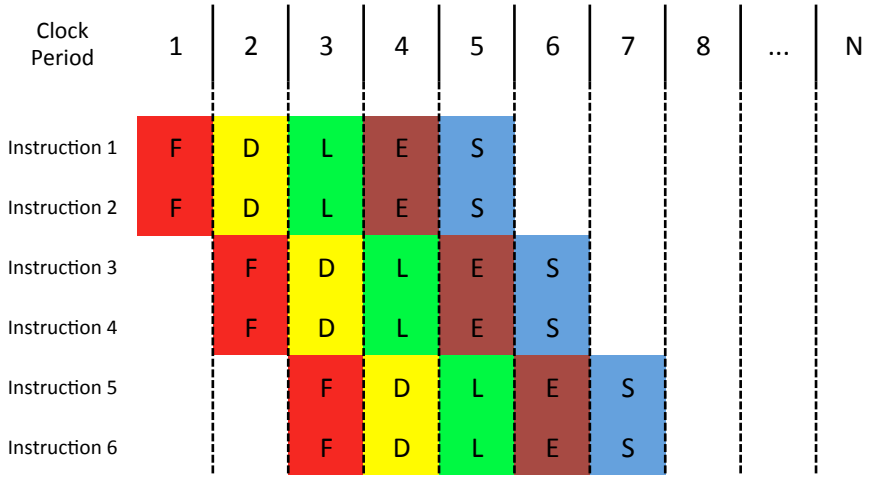
The AMD Opteron 6380 Abu Dhabi 2.5GHz



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

To the Extreme - Parallel Inside

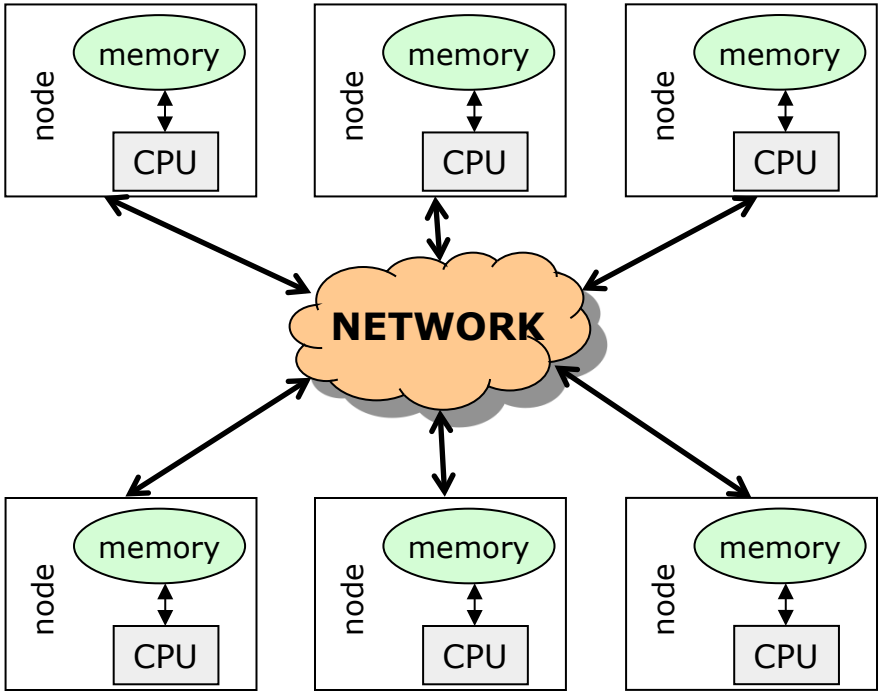
Vector Units for processing multiple data in //



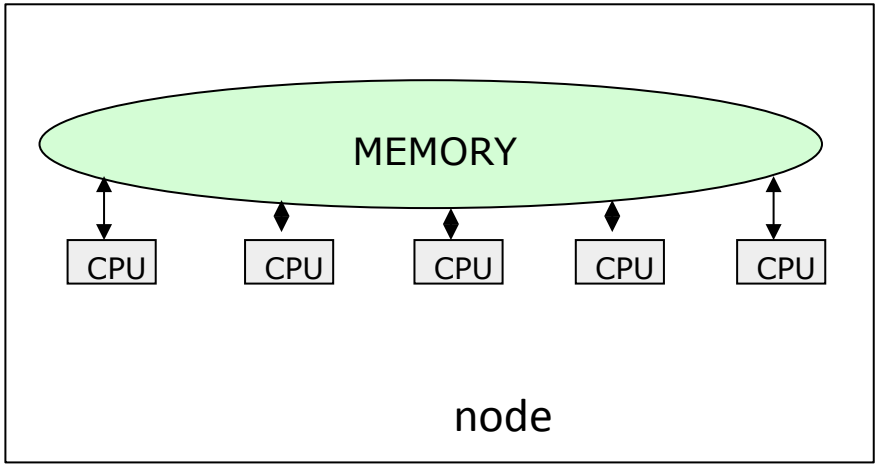
Pipelined/Superscalar design: multiple functional units operate concurrently

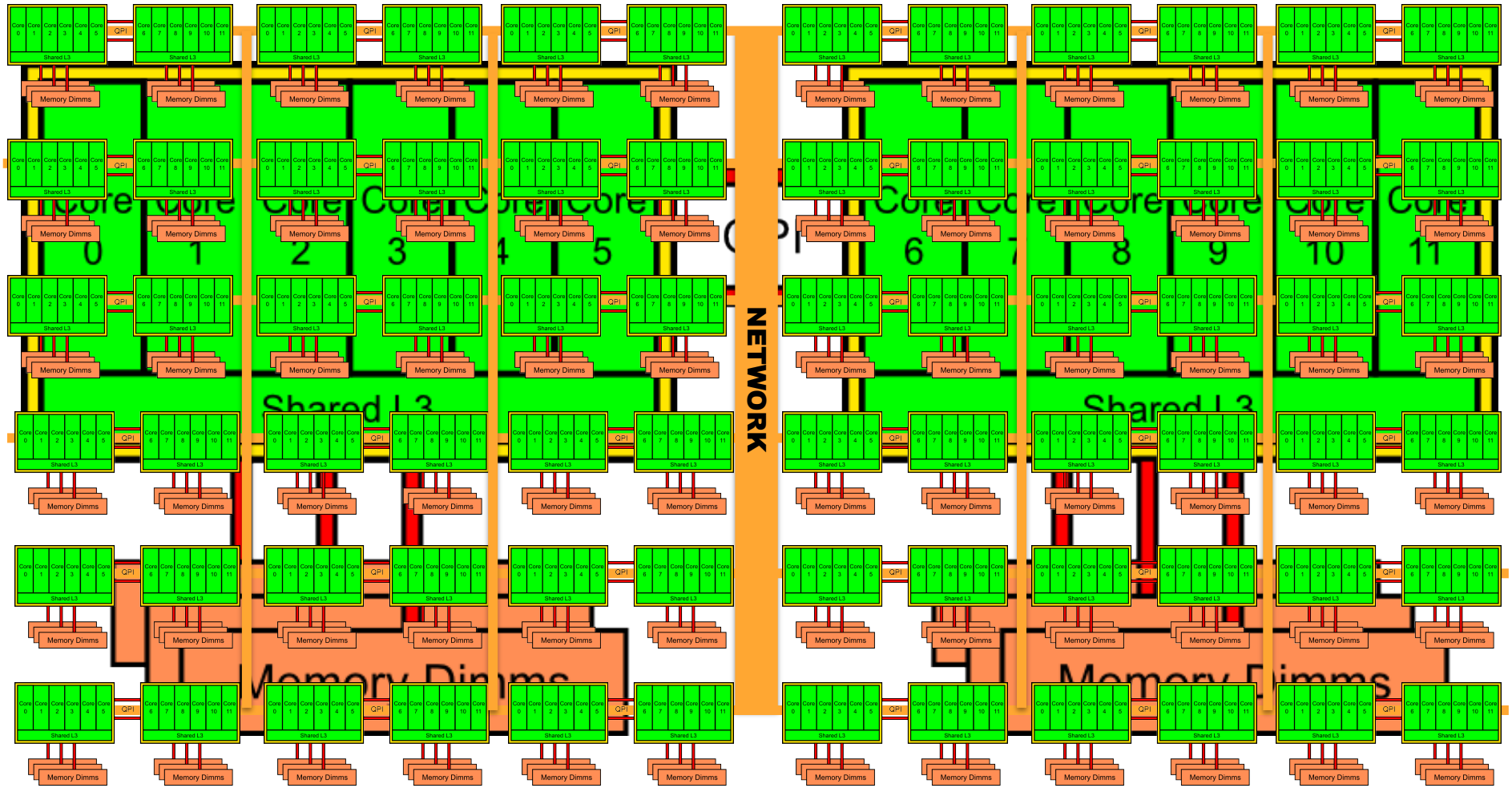
Parallel Architectures

- Distributed Memory



- Shared Memory




NETWORK
Memory Dimms
Shared L3
QPI
Core 0
Core 1
Core 2
Core 3
Core 4
Core 5
Core 6
Core 7
Core 8
Core 9
Core 10
Core 11

FOUNDATION OF PARALLELISM

Principles of Parallel Applications

- 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!!!

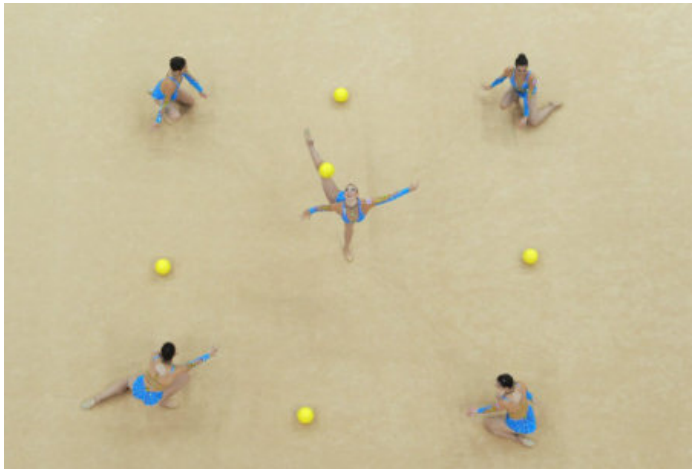
Type of Parallelism

- **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

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**)



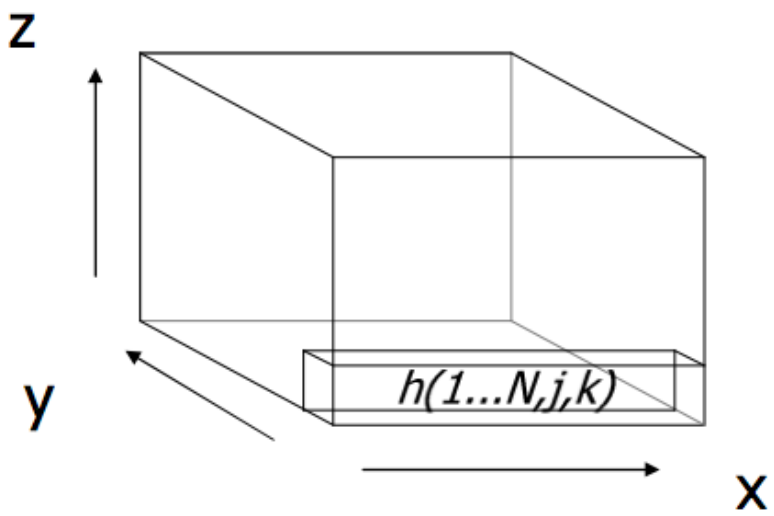
Programming Parallel Paradigms

- 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

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
- OpenMP (threading)
  - It is a defined standard for programming shared memory architecture
  - A single process (i.e., MPI process) spawns sub-processes (threads)
  - Communication take places in memory: available only for shared mem. architecture
  - Achieved with compiler directives and/or via call to multi-threading libraries
- Quantum ESPRESSO exploits both MPI and OpenMP parallelization.

Case Study: 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

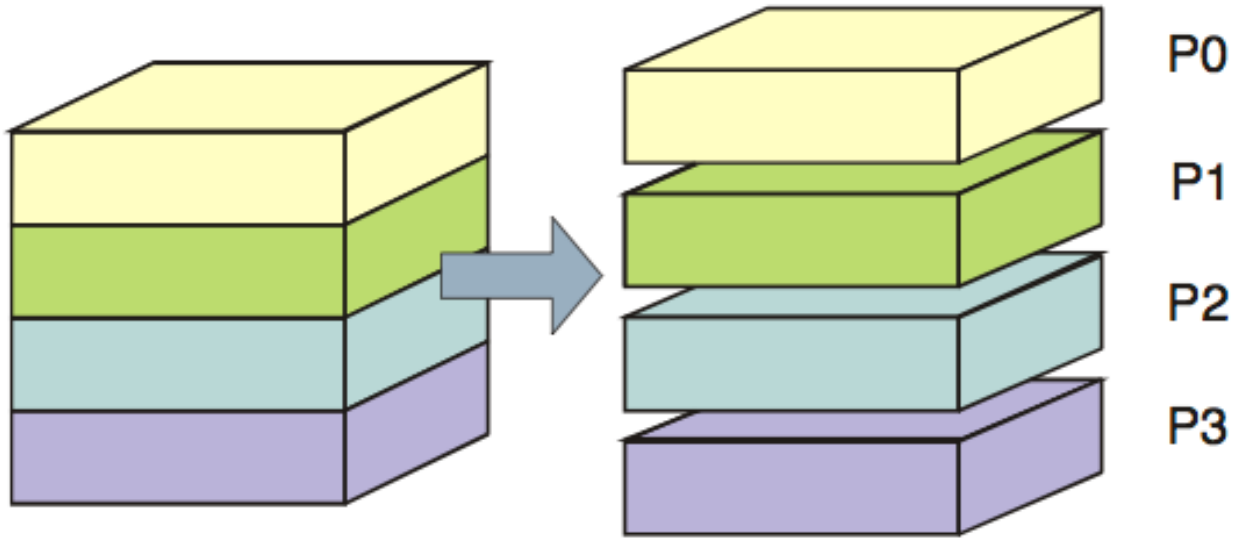
DFT long z-dimension

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

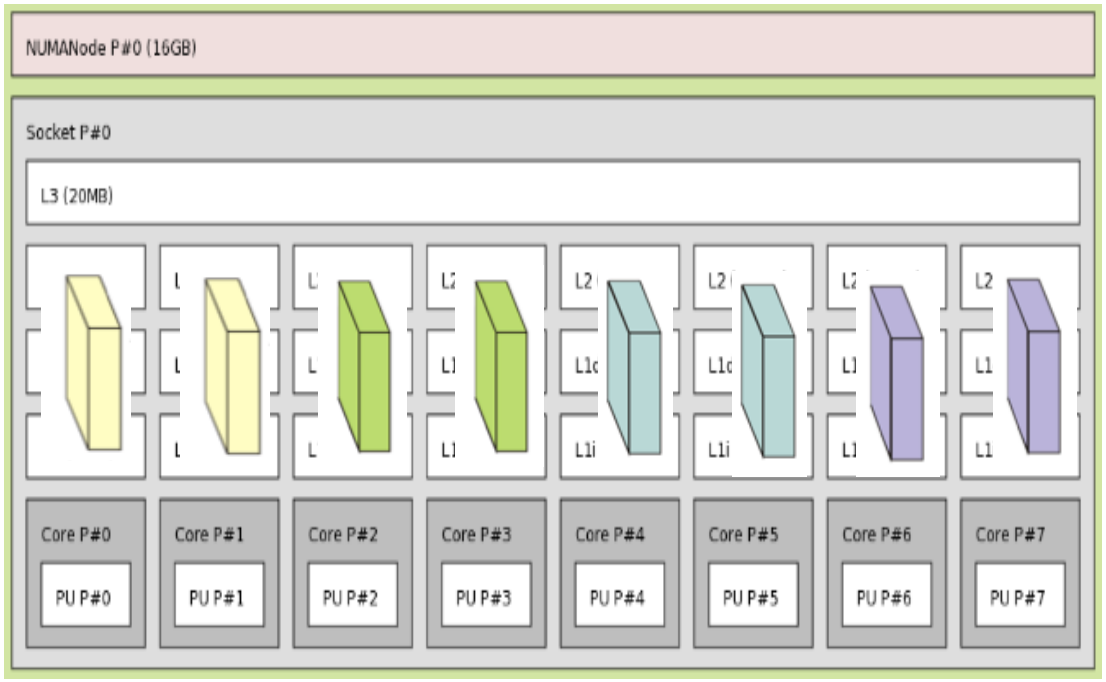
2) For any value of ***i*** and ***k*** transform the column (i, 1…N, k)

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

Parallel 3DFFT / 1

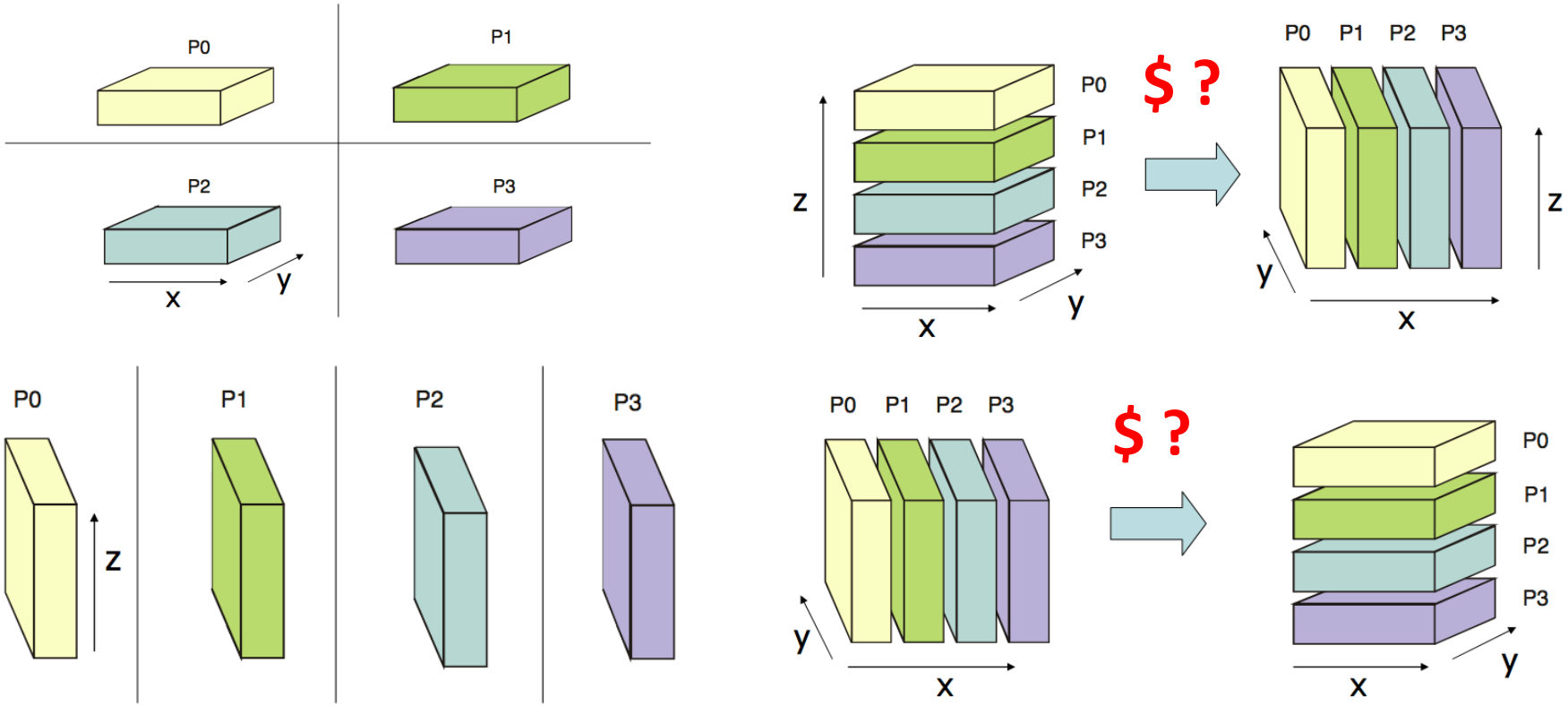


Parallel 3DFFT on Parallel Systems



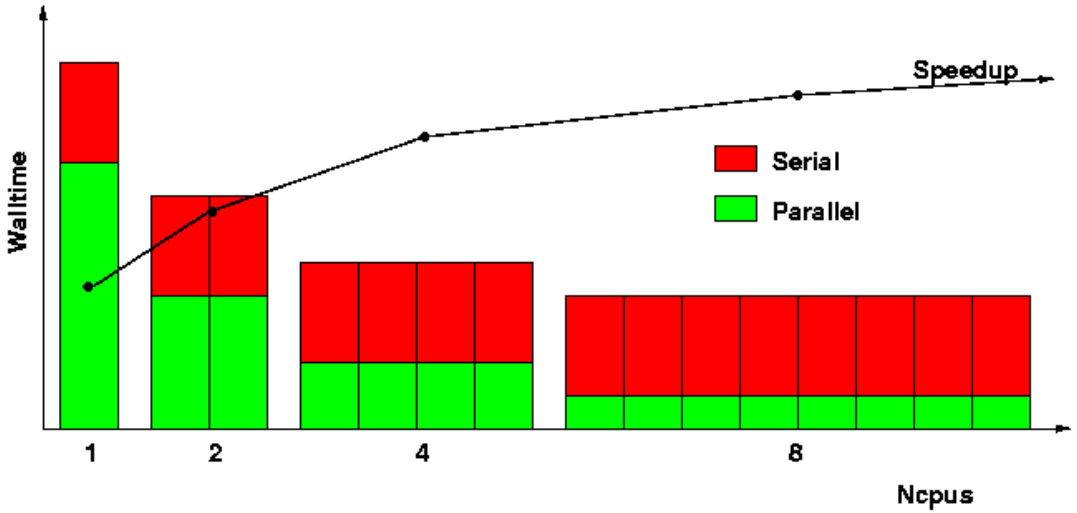
The Intel Xeon E5-2665
Sandy Bridge-EP 2.4GHz

Parallel 3DFFT / 2



Amdahl's law

In a massively parallel context, an upper limit for the scalability of parallel applications is determined by the fraction of the overall execution time spent in non-scalable operations (Amdahl's law).



maximum speedup tends to
$1 / (1 - P)$
P= parallel fraction

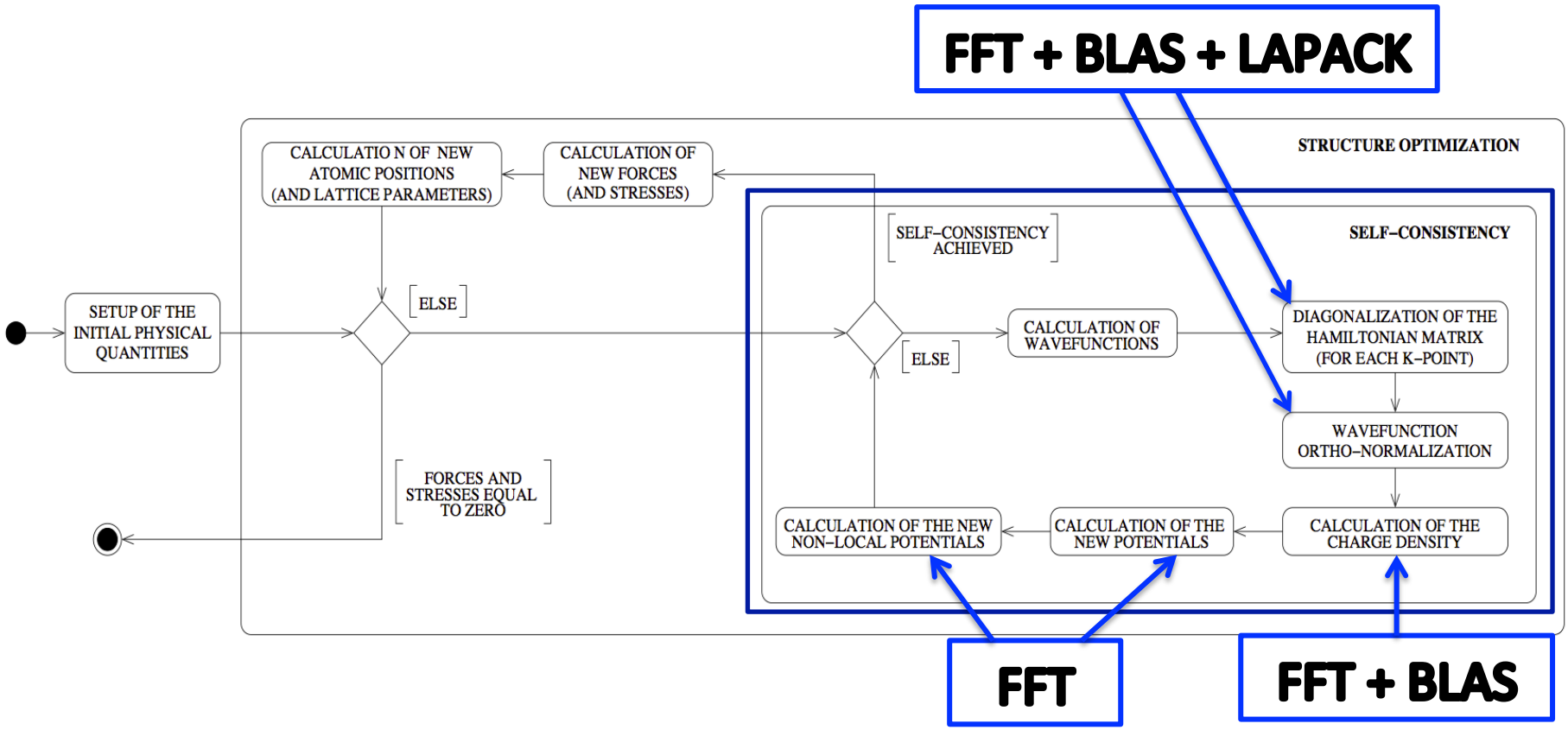
1000000 core

P = 0.999999

serial fraction= 0.000001

A case study: QE-PWscf

HIGH-PERFORMANCE AND PARALLEL COMPUTING IN THE QE DISTRIBUTION



* Spiga, F. & Girotto, I. phiGEMM: A CPU-GPU Library for Porting Quantum ESPRESSO on Hybrid Systems, 10.1109/PDP.2012.72 Publication Year: 2012 , Page(s): 368 - 375. IEEE Conference Publications

Master the horsepower

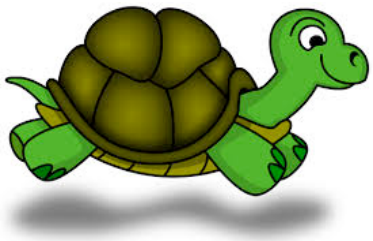
- Spend the effort to understand the analyze the computer platform as well as the software environment
  - documentation, ask to your sys-admin
- Use appropriate libraries, compilers and optimizations
- configure && make
  - it is for a portable package and general purpose application, not for HPC!!!
- The best tuning starts from the file make.sys

Compiling/Linking QE

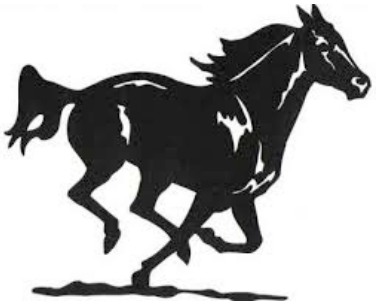
- Environment Loading
  - module load profile/advanced
  - module load bgq-xl/1.0
  - module load essl/5.1 lapack/3.4.1--bgq-xl--1.0 scalapack/2.0.2--bgq-xl--1.0 mass/7.3--bgq-xl--1.0
- Configure
  - $ ./configure --enable-parallel --enable-openmp arch=ppc64-bg --disable-shared
- *make.sys*
  - FDFLAGS = -D__XLF,-D__FFTW,-D__ESSL,-D__LINUX_ESSL,-D__MASS,-D__MPI,-D__PARA,-D__OPENMP,-D__BGQ,-D__SCALAPACK
  - BLAS_LIBS = -L/opt/ibmmath/essl/5.1/lib64/ -lesslsmpbg
  - LAPACK_LIBS = -L/cineca/prod/libraries/lapack/3.4.1/bgq-xl--1.0/lib -llapack
  - SCALAPACK_LIBS = -L/cineca/prod/libraries/scalapack/2.0.2/bgq-xl--1.0/lib -lscalapack
  - MASS_LIBS = -lmassv -lmass_simd

* useful reference in the Glenn K. Lockwood (SDSC) Blog

Libraries



Internal QE



Freely available
Open Source Optimized

Third-Party Highly-Optimized

netlib

FFTW MAGMA ATLAS

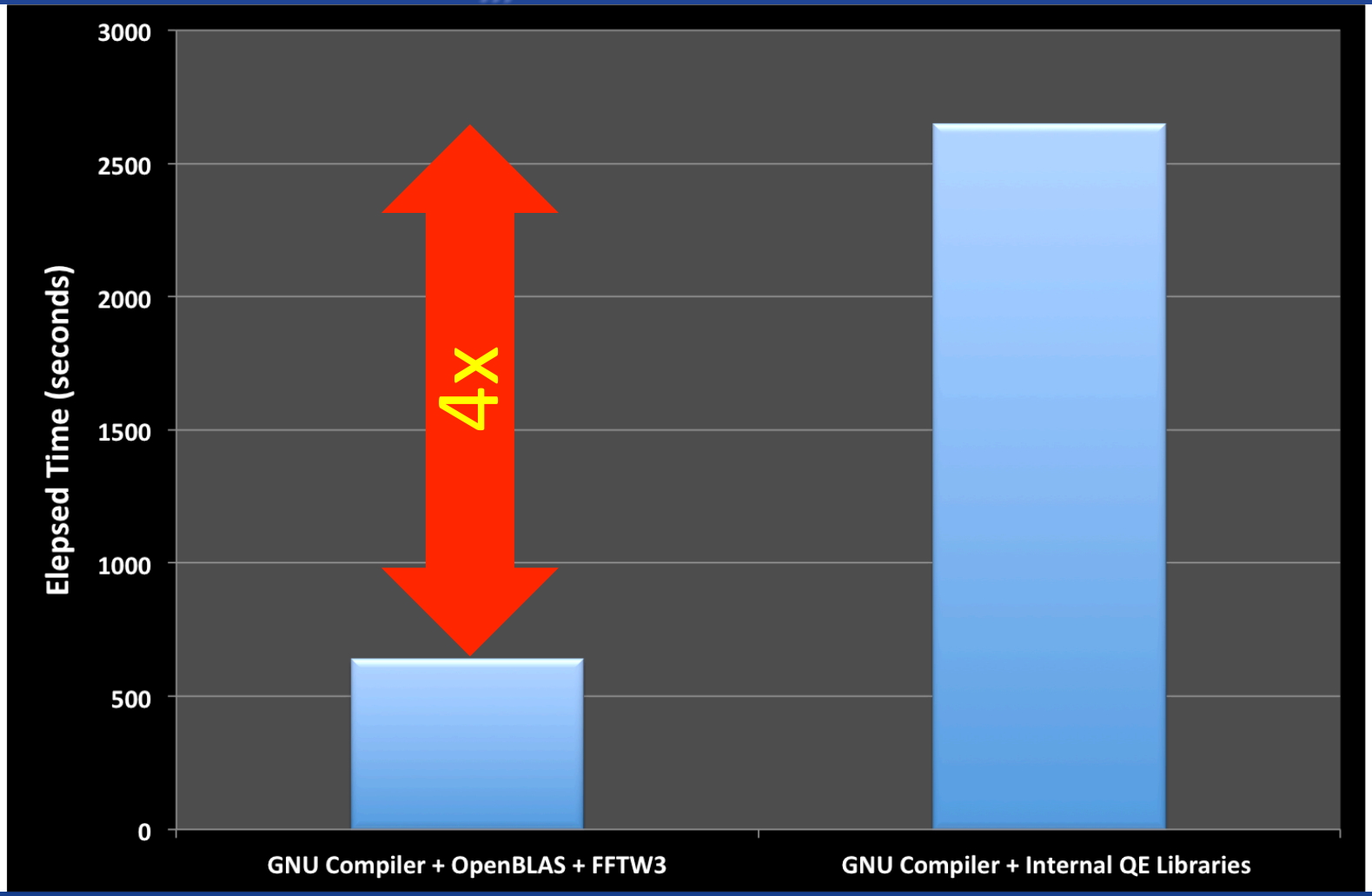
OpenBLAS PLASMA

AMD Core Math Library ACML

Intel Math Kernel Library

nag Results Matter. Trust NAG.

IBM ESSL


3000
2500
2000
1500
1000
500
0
Elepsed Time (seconds)
4x
GNU Compiler + OpenBLAS + FFTW3
GNU Compiler + Internal QE Libraries

SCF:
- compute potential
- **solve KS eigen-problem**
  - Loop over k-points:
    - **Davidson iteration / CG iteration:**
      - compute/update H * psi:
        - compute kinetic and non-local term (in G space)
        - Loop over (not converged) bands:
          - FFT psi to R space
          - compute V * psi
          - FFT V * psi back to G space
        - compute EXX:
          - ....
      - project H in the reduced space (ZGEMM)
      - diagonalize the reduced Hamiltonian:
        - cholesky factorization
        - call to LAPACK/SCALAPACK diagonalization routine
  - compute new density
    - loop over k-points:
      - loop over bands:
        - FFT psi to R space
        - accumulate psi
    - charge density symmetrisation

$$\hat{H}_{KS}\left|\psi_{\vec{k},b}\right\rangle$$

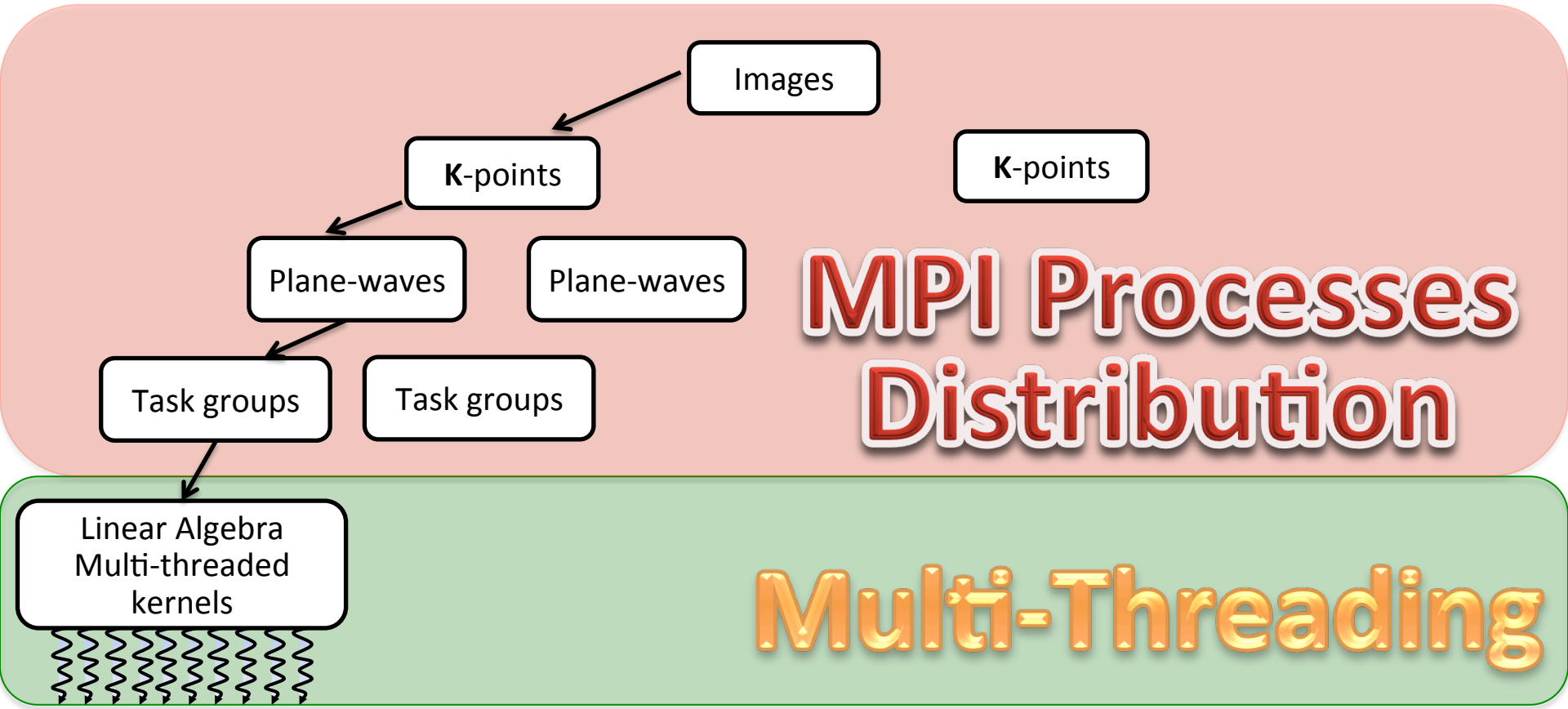
$$\left\langle \vec{k}+\vec{G}\left|\hat{H}_{KS}\right|\vec{k}+\vec{G}'\right\rangle$$

$$n(\vec{r}) = 2\sum_{k}\sum_{v}\left|\psi_{v,\vec{k}}(\vec{r})\right|^2$$

$$\hat{H}_{KS}\left|\psi_{\vec{k},b}\right\rangle = \varepsilon_{\vec{k},b}\left|\psi_{\vec{k},b}\right\rangle$$

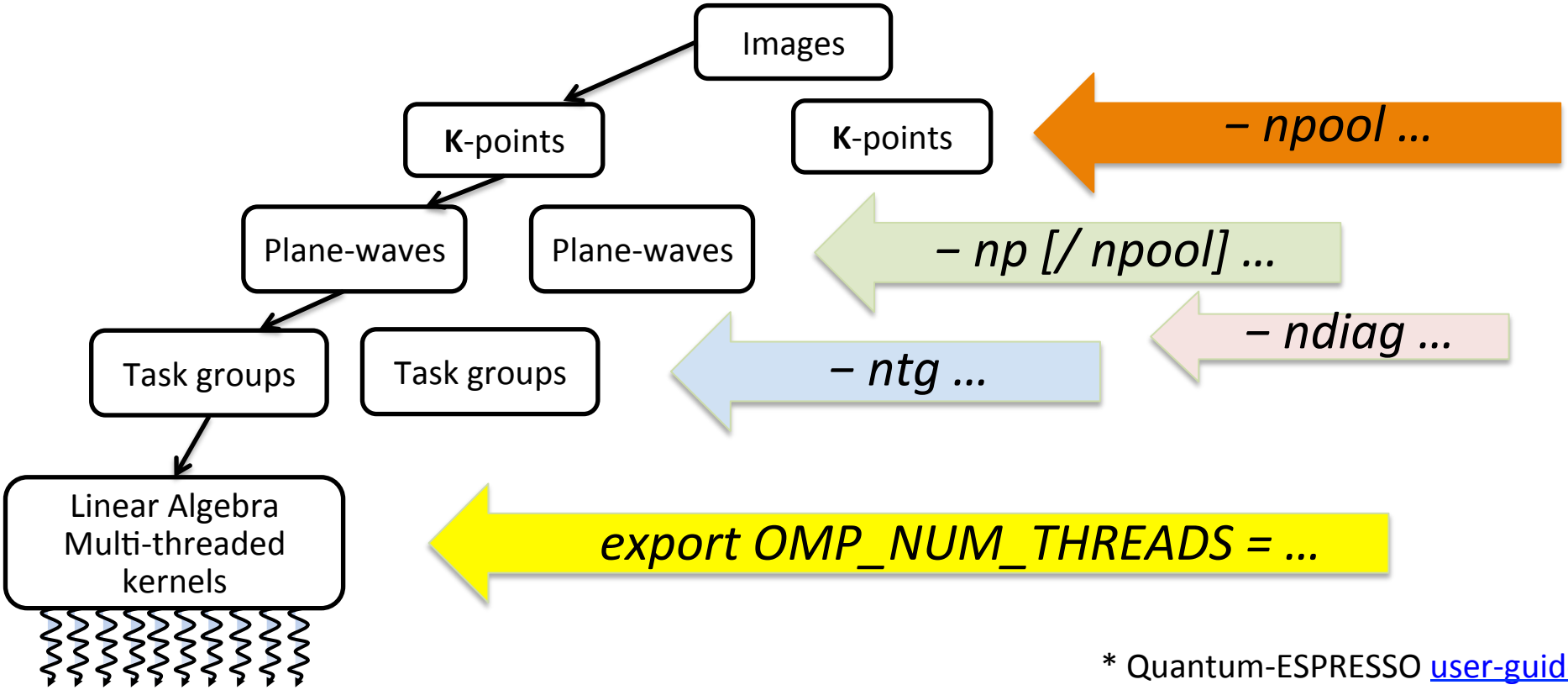
Courtesy of F.Spiga & P.Giannozzi

Levels of parallelism in pw.x



* Giannozzi P. & Cavazzoni C. 2009 C 32 at press doi:10.1393/ncc/i2009-10368-9

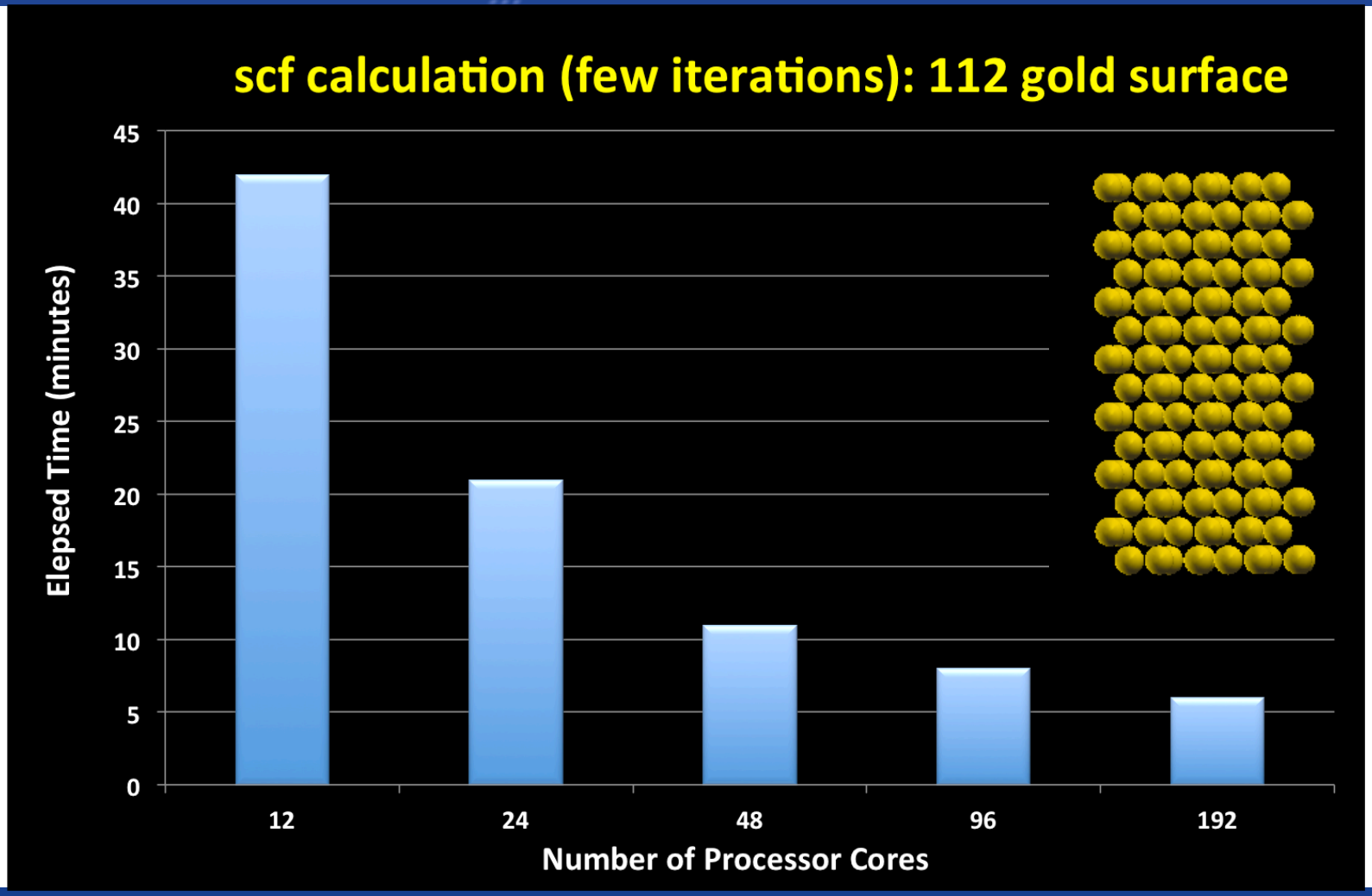
Levels of parallelism in pw.x



* Quantum-ESPRESSO user-guide

Plane-Wave parallelization

- Wave-function coefficients are distributed among processes that each MPI process works on a subset of plane waves. The same is done on real-space grid points.
- It is the default parallelization scheme which is nested into other higher levels of parallelism (if specified): k-points, images, etc...
- Plane-wave parallelization is historically the first serious and effective parallelization of PW-PP codes as well as for the QE distribution
- Good memory distribution and load-balance
- Heavy and frequent intra-CPU communications in the 3D FFT
- Limited in scalability by N3 (3^{rd} dimension of charge density 3DFFT grid)


scf calculation (few iterations): 112 gold surface
Elepsed Time (minutes)
45
40
35
30
25
20
15
10
5
0
12
24
48
96
192
Number of Processor Cores

Key-point parallelization

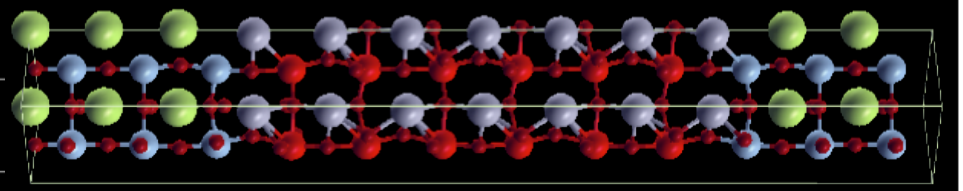
- k-points are computationally considered as independent entities distributed among (n)pools of cores: computationally intensive operations are performed within a pool and concurrently to others pools
- Communication among pools is required only in order to sum local contributions: charge density, and in computing the total energies, total ionic forces, etc...
- Limited by number of k-points: npools must be a divisor of the number of cores involved in the computation and of the overall number of k-points (so that they are equally distributed)

```
mpirun -np 16 pw.x -npool 4 -inp input file
```

* L.J. Clarke, I. Stich, M.C. Payne, Comput, Phys. Commun. 72 (1992) 14

scf calculation (few iterations): LSMO BiFeO3 (BFO)

magnetic heterostructure (120 atoms) studied with 8 k-points

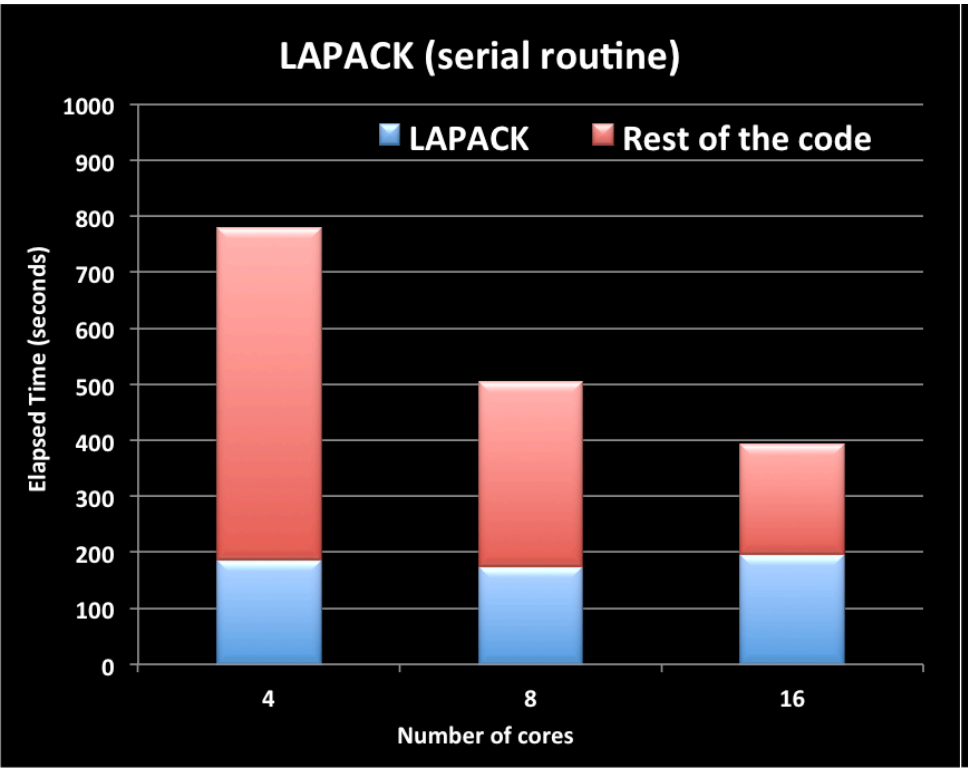


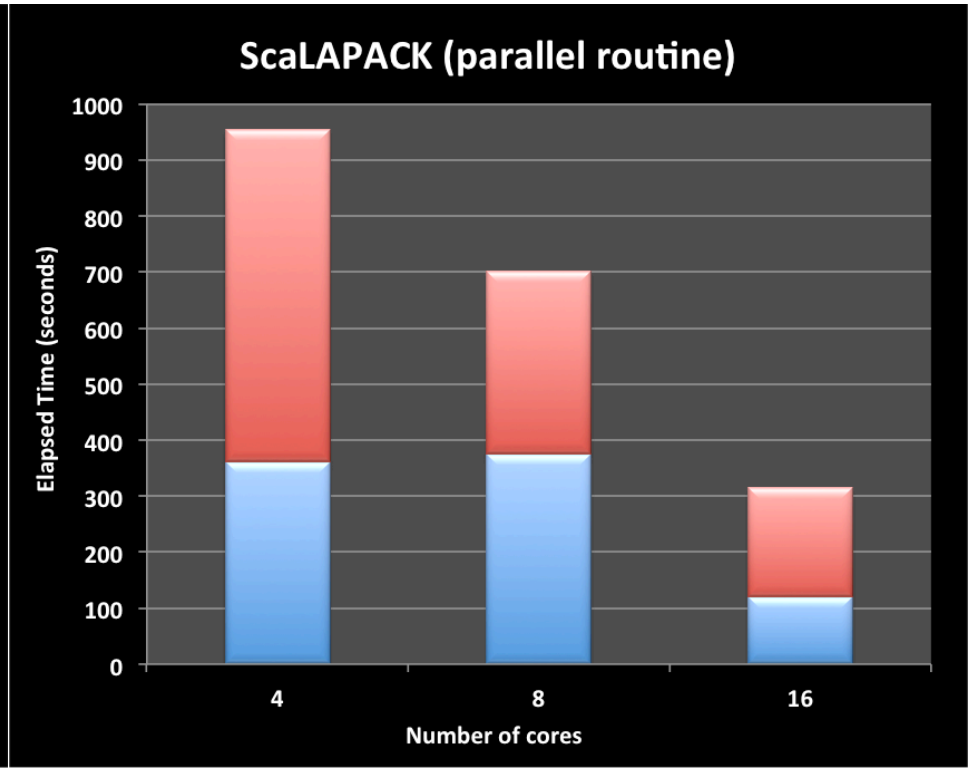
Courtesy of Rodrigo Neumann Barros Ferreira, PhD Student Solid State Physics Department – Physics Institute, Rio de Janeiro Federal University

Linear-Algebra parallelization

- Distribute and parallelize matrix diagonalization and matrix- matrix multiplications needed in iterative diagonalization (SCF) or orthonormalization (CP). Introduces a linear-algebra group of ndiag processors as a subset of the processors involved within the plane-wave parallelism.
- It is requested to be a square number
- RAM is efficiently distributed: removes a major bottleneck in parallel systems
  - improves speedup in large systems
- Scaling tends to saturate: testing required to choose optimal value of ndiag
- If compiled with ScaLAPACK the code wiil try to automatically set this number. Not highly optimized. If no ScaLAPACK a serial LAPACK call is used by default. If the ndiag is specified with no ScaLAPACK an home-made parallel algorithms are used. Useful in supercells containing many atoms, where RAM and CPU requirements of linear- algebra operations become a sizable share of the total.

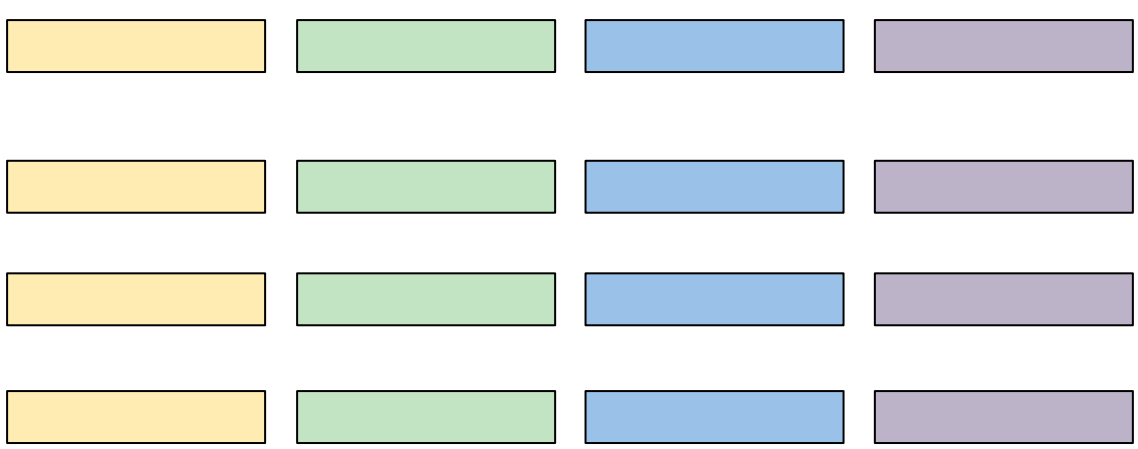
* Cavazzoni C . and Chiarotti G .L 1999 Comput. Phys. Commun. 123 56

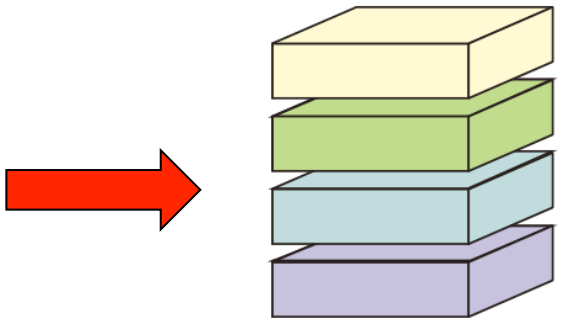

LAPACK (serial routine)
LAPACK
Rest of the code
Elapsed Time (seconds)
1000
900
800
700
600
500
400
300
200
100
0
4
8
16
Number of cores


ScaLAPACK (parallel routine)
Elapsed Time (seconds)
1000
900
800
700
600
500
400
300
200
100
0
4
8
16
Number of cores

Task Group Parallelism

- Think at a 2D domain problem





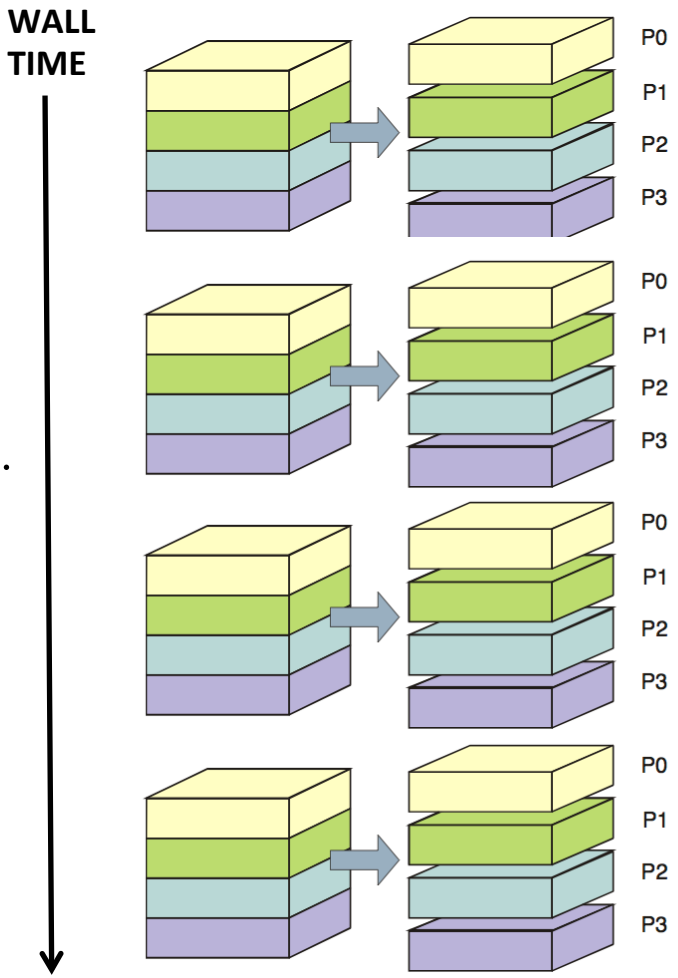
pis(i)

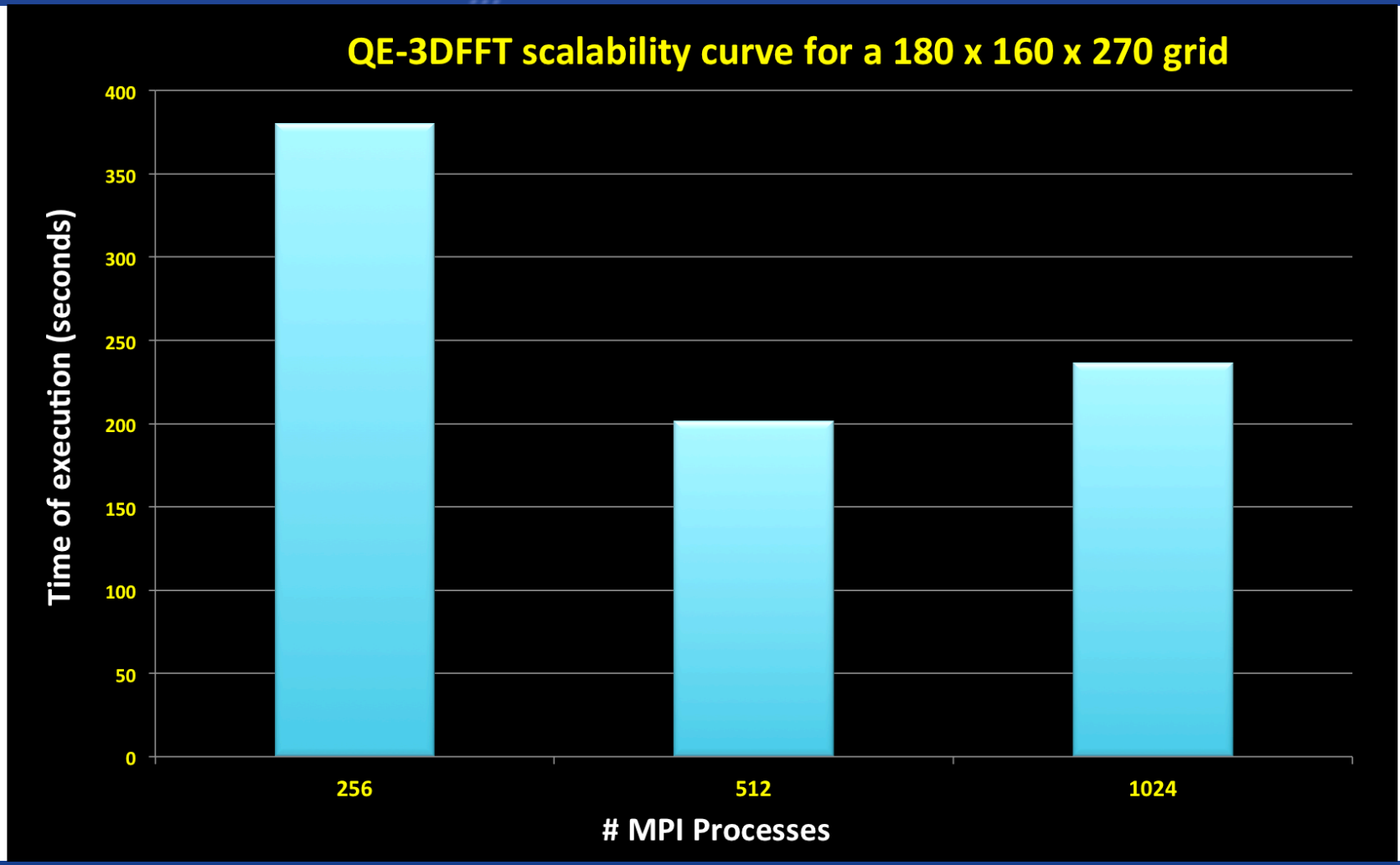
```
do i = 1, n
   compute 3D FFT( psi(i) )
end do
```

* Hutter J and Curioni A 2005 Car-Parrinello molecular dynamics on massive parallel computers ChemPhysChem 6 1788

Best Practice: 3DFFT tuning

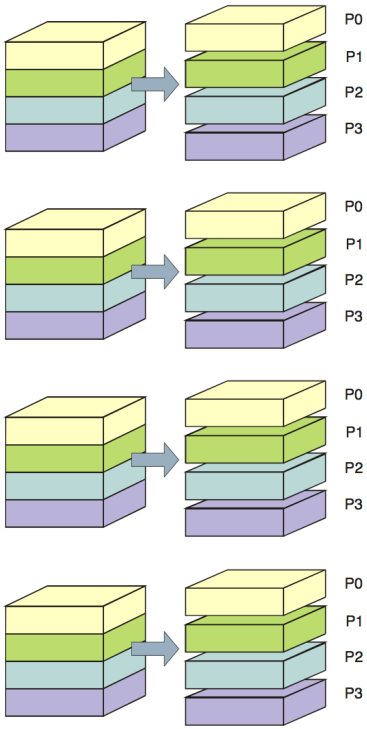
```
mpirun -np 4 … pw.x -ntg 1 …
```




QE-3DFFT scalability curve for a 180 x 160 x 270 grid
Time of execution (seconds)
400
350
300
250
200
150
100
50
0
256
512
1024
MPI Processes

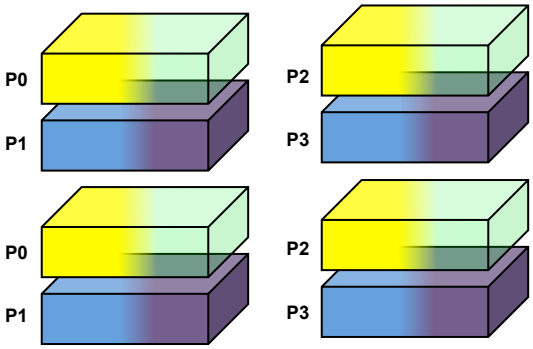
Task Group Parallelism

WALL TIME



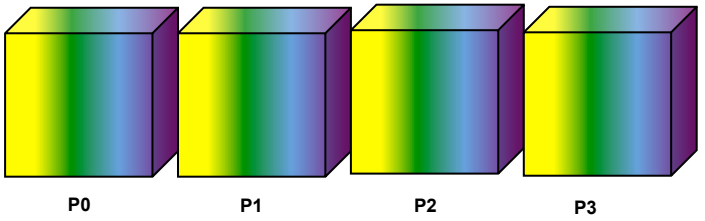
`mpirun -np 4 … pw.x -ntg 1 …`

Merge Distributed Data

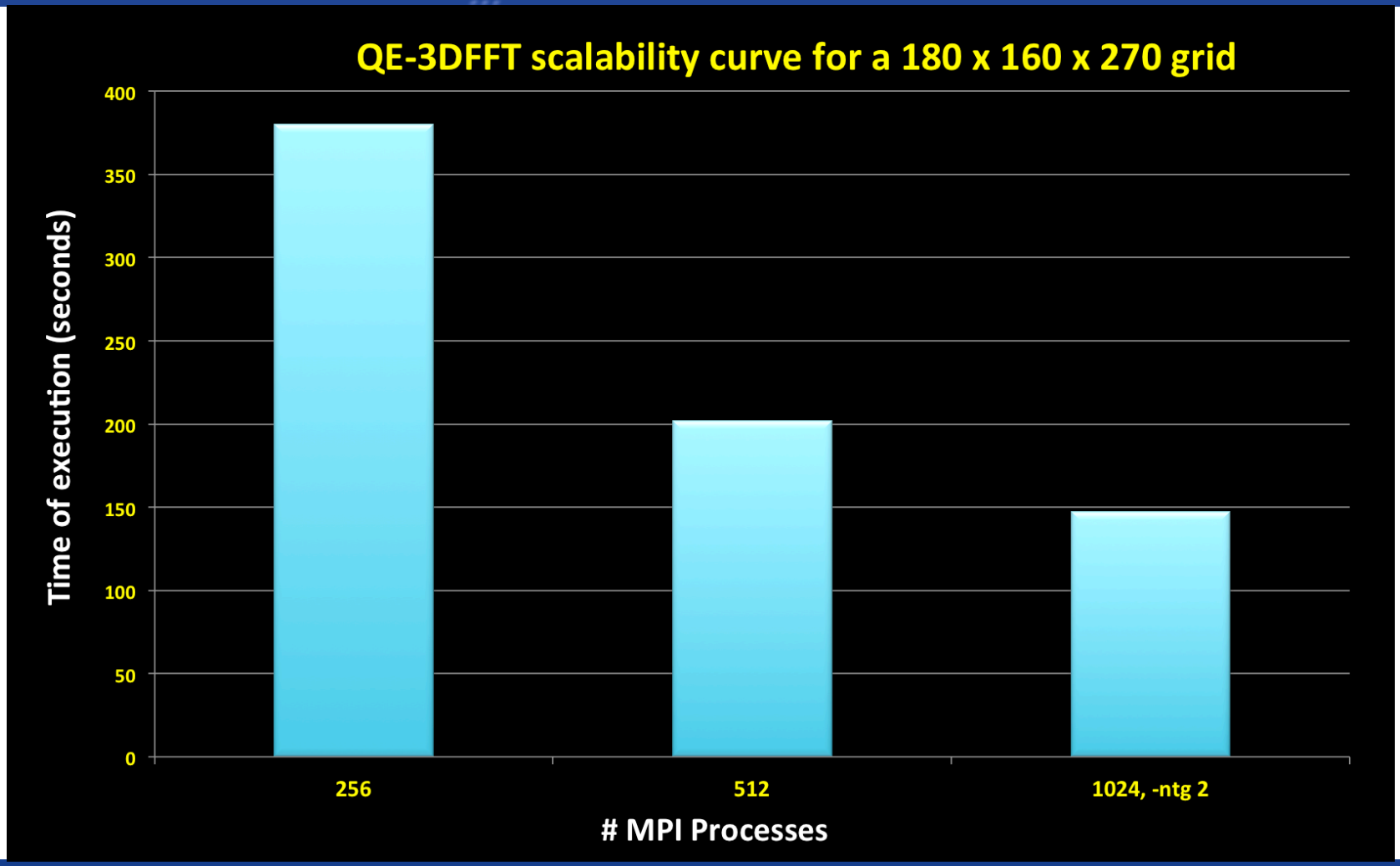


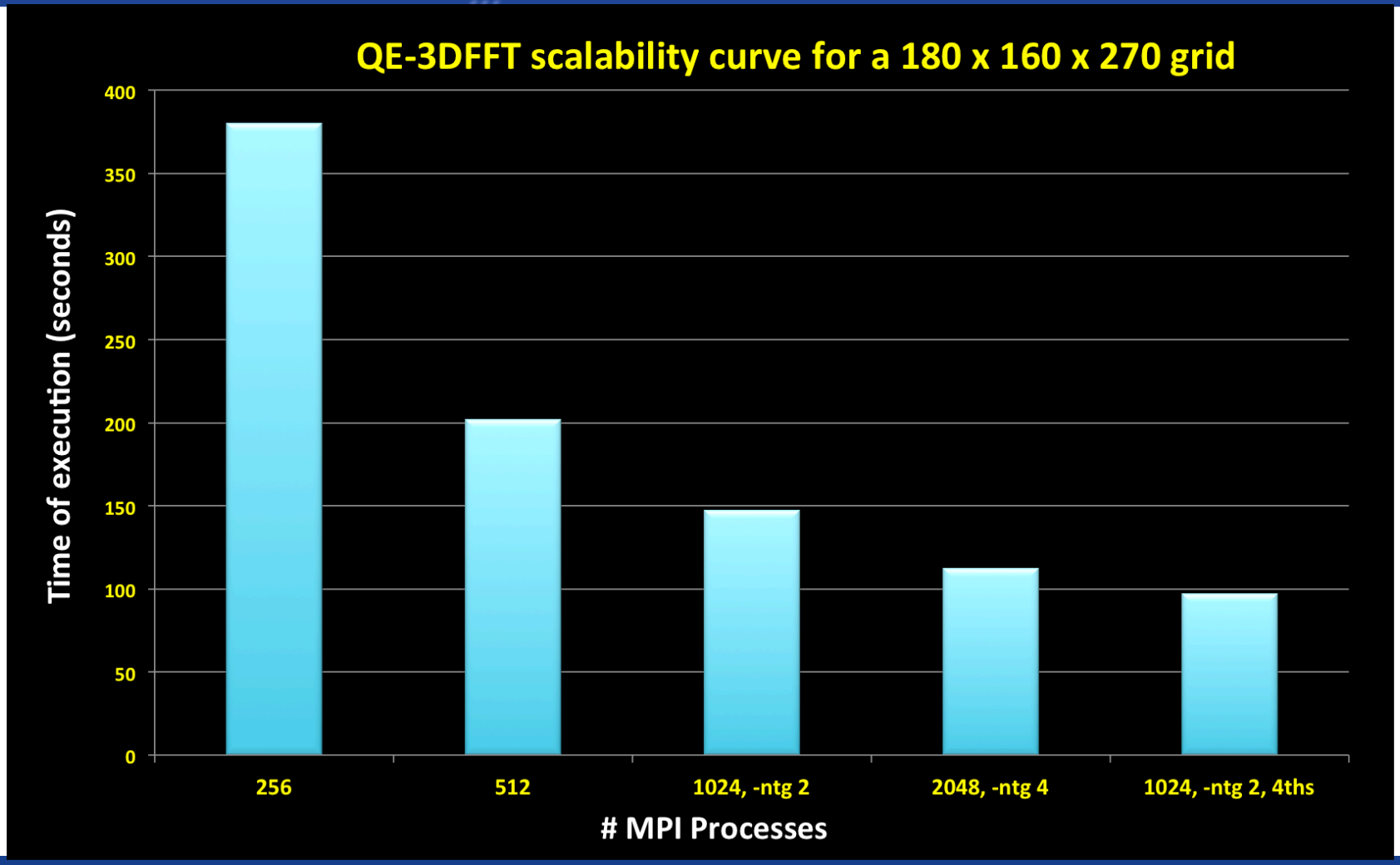
`mpirun -np 4 … pw.x -ntg 2 …`

Merge Distributed Data



`mpirun -np 4 … pw.x -ntg 4 …`


QE-3DFFT scalability curve for a 180 x 160 x 270 grid
Time of execution (seconds)
400
350
300
250
200
150
100
50
0
256
512
1024, -ntg 2
MPI Processes


QE-3DFFT scalability curve for a 180 x 160 x 270 grid
Time of execution (seconds)
400
350
300
250
200
150
100
50
0
256
512
1024, -ntg 2
2048, -ntg 4
1024, -ntg 2, 4ths
MPI Processes

Conclusions

- Calculation of non trivial system with PW codes requires a decent technological background:
  - to avoid huge waste of time
  - to make possible studying complex system
- QE is an high-performance code if used efficiently
- The complexity of the code is bound to increase in the era of the multi- and many-cores architectures

Thanks for your attention!!

Acknowledgements:

- **Filippo Spiga (Cambridge University/QE-Foundation)**
- Paolo Giannozzi (Udine University)
- Carlo Cavazzoni (CINECA)
- Layla Martin-Samos (University of Nova Gorica)
- Mike Atambo (ICTP)