The Abdus Salam
International Centre
for Theoretical Physics

How to Evaluate The Efficiency of The Parallelism

Ivan Girotto – igirotto@ictp.it
Information & Communication Technology Section (ICTS)
International Centre for Theoretical Physics (ICTP)

Reminder

October 23, 2012 – 11AM – 12:15 Location: Leonardo Da Vinci Building, Euler Lecture Hall (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 (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 (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.

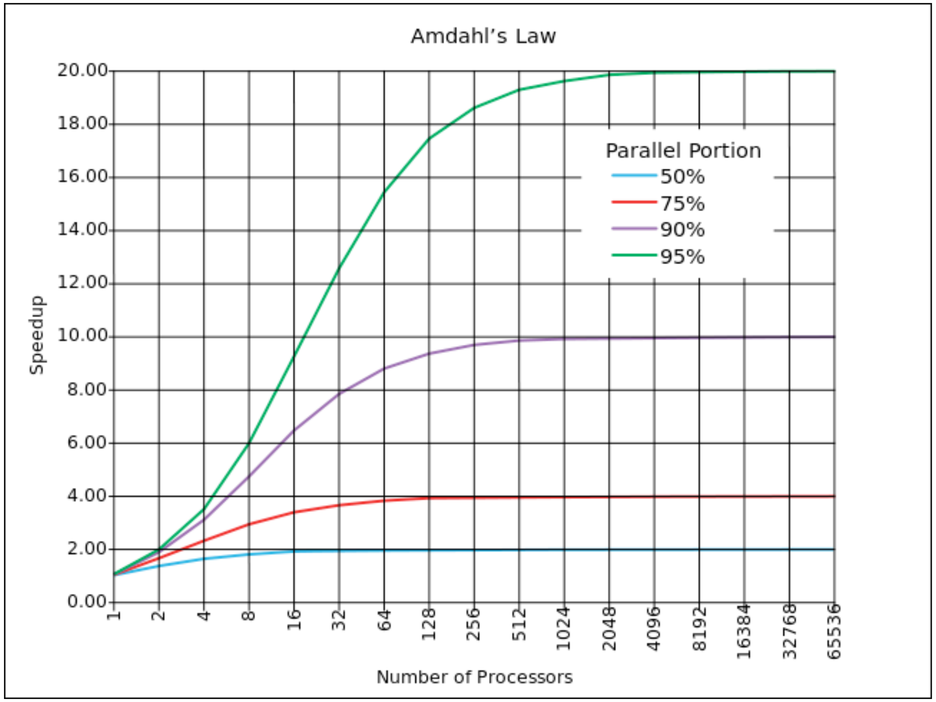
Outline

- Recap of the Previous Appointment
- Speedup and Efficiency
- The Fascinating Concept of Scalability
- Some Examples and Good Practices
- Conclusions

The Previous Appointment

- Technological Change
- Parallelism and concurrency
- Type of parallelism
- Granularity

The Amdahl's Law



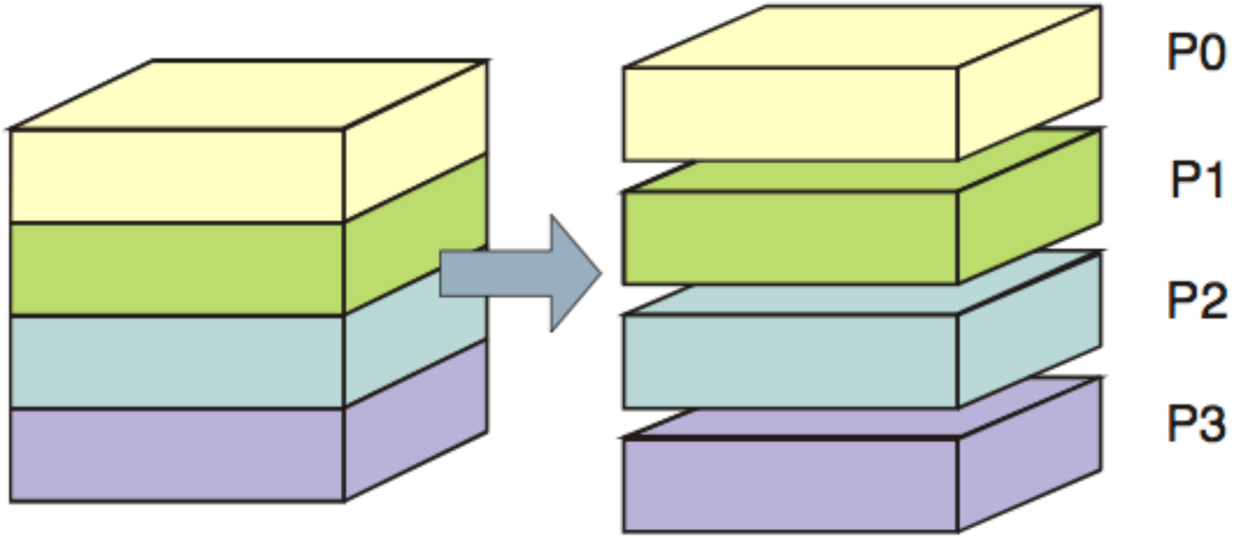
The speedup of a program using multiple processors in parallel computing is limited by the sequential fraction of the program. For example, if 95% of the program can be parallelized, the theoretical maximum speedup using parallel computing would be 20× as shown in the diagram, no matter how many processors are used.

… but we know this is not the whole story!! ☹

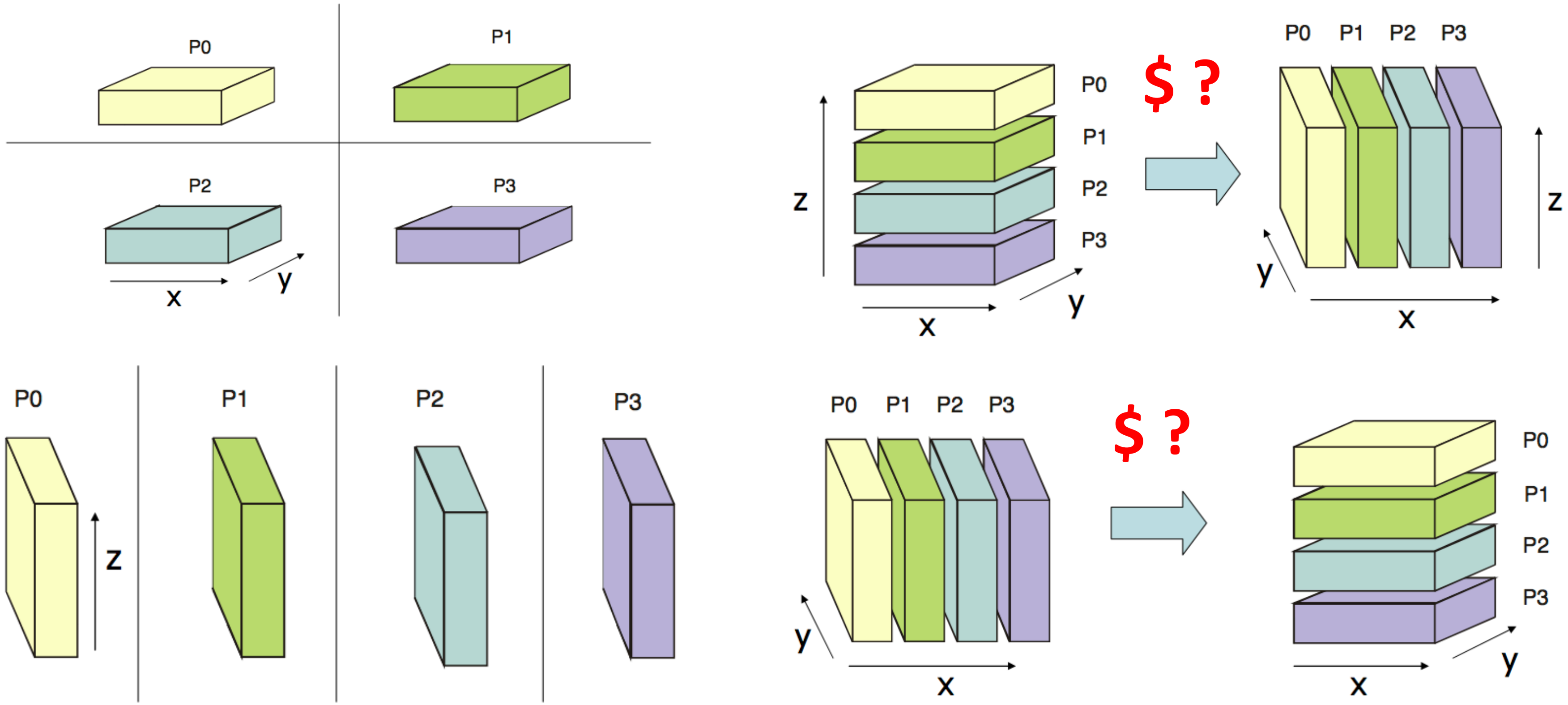
Few basic points

- A sequential algorithm is usually evaluated in term of execution time => problem size (O)
- // algorithm depends also on the number of processing elements used and the overhead that the parallelism introduces
- Parallel algorithm/problem/program can be hardly evaluated without considering the parallel architecture

Parallel 3DFFT / 1

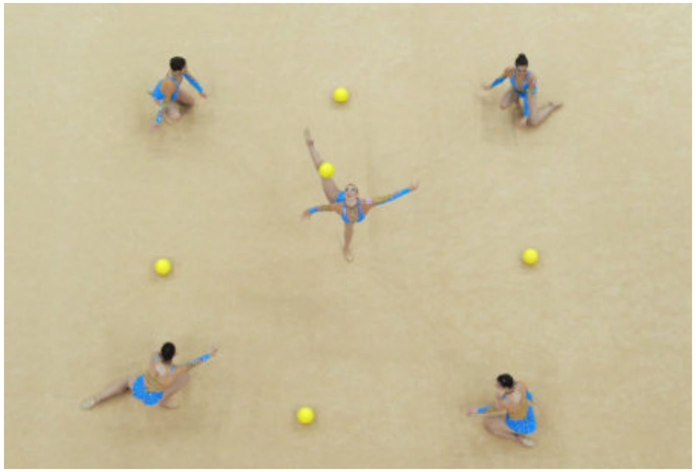


Parallel 3DFFT / 2

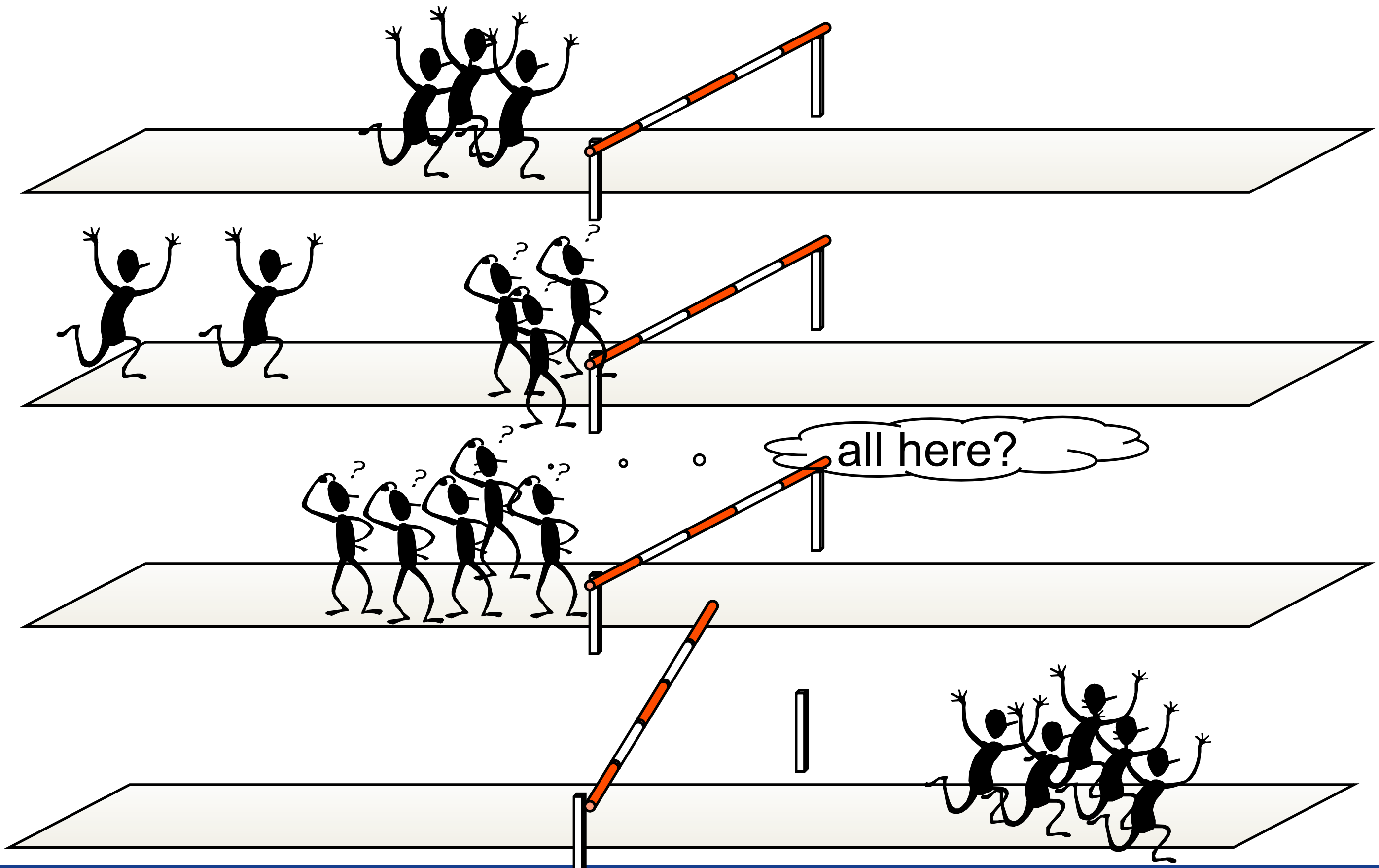


Process Interactions

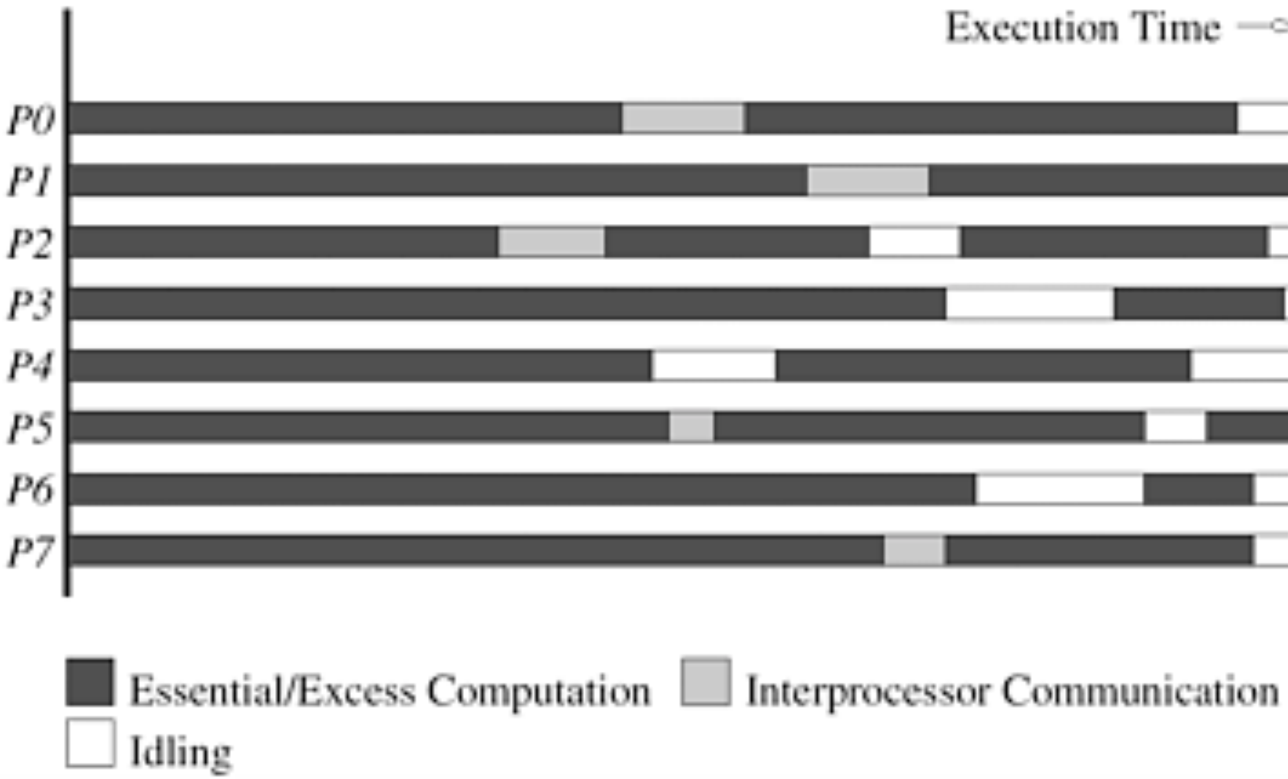
- The effective improvement obtained by the parallelization depend by the amount of overhead we introduce making the algorithm parallel
- There are mainly two key sources of overhead:
    1. Time spent in inter-process interactions (**communication**)
    2. Time some process may spent being idle (**synchronization**)



What happens if the girls are not well trained?



The execution profile of a hypothetical // program

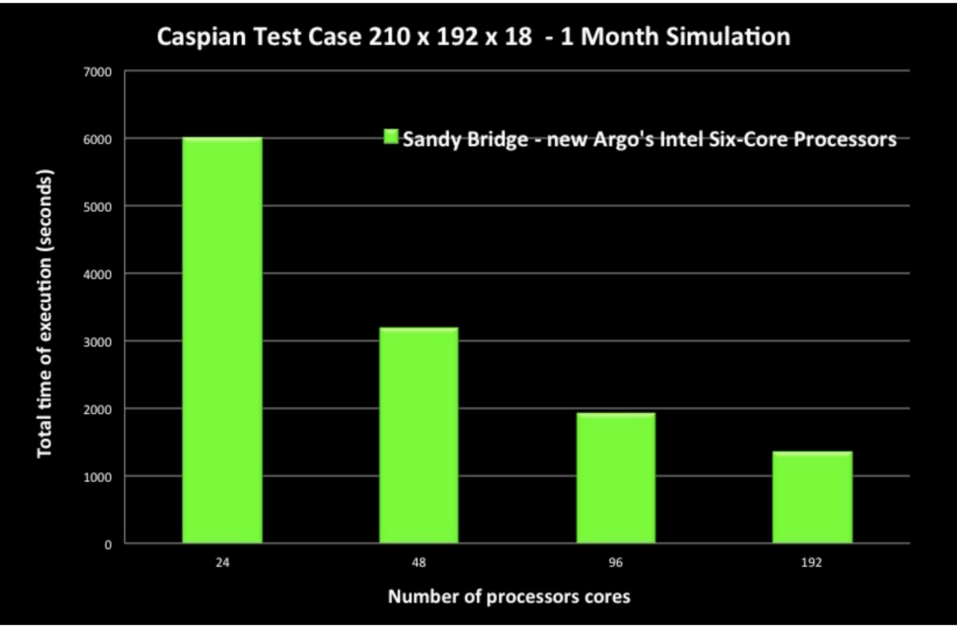


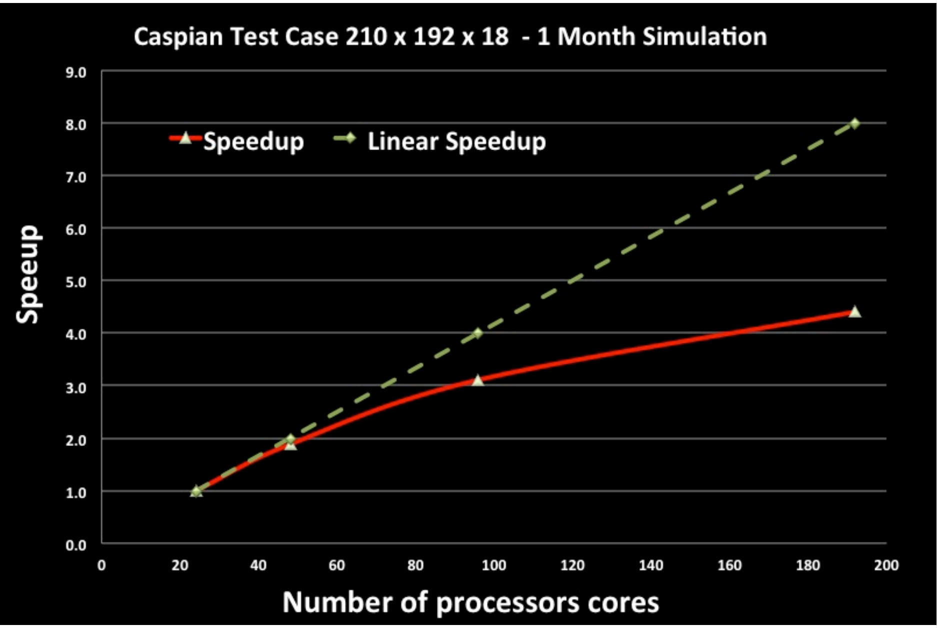
How do we evaluate the improvement?

- We want estimate the amount of the introduced overhead => $T_o = n_{pes} T_P - T_S$
- But to quantify the improvement we use the term **Speedup**:

$$S_P = \frac{T_S}{T_P}$$

Speedup


Caspian Test Case 210 x 192 x 18 - 1 Month Simulation
Sandy Bridge - new Argo's Intel Six-Core Processors
Total time of execution (seconds)
0
1000
2000
3000
4000
5000
6000
7000
24
48
96
192
Number of processors cores

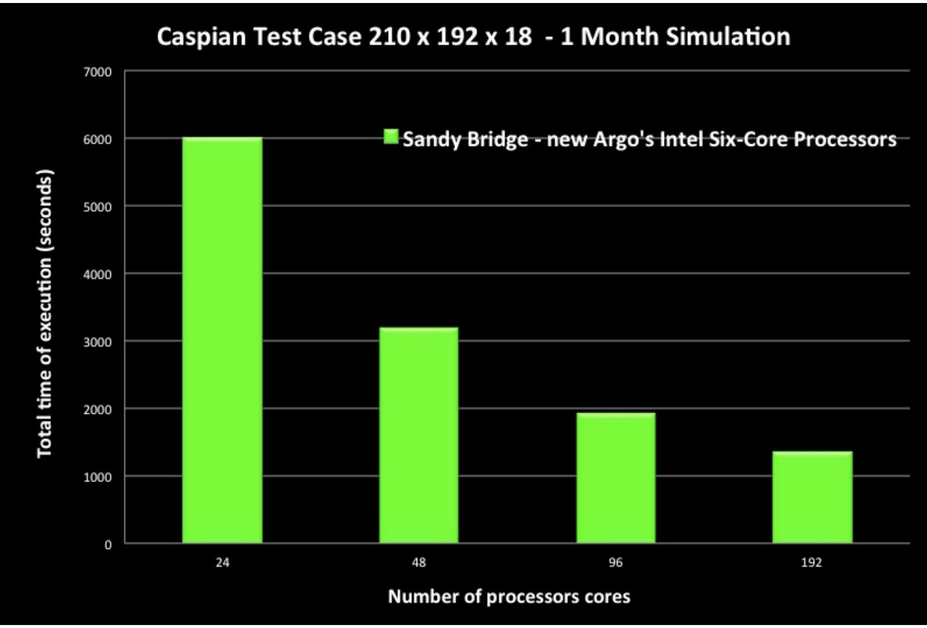

Caspian Test Case 210 x 192 x 18 - 1 Month Simulation
Speedup
Linear Speedup
Speeup
0.0
1.0
2.0
3.0
4.0
5.0
6.0
7.0
8.0
9.0
0
20
40
60
80
100
120
140
160
180
200
Number of processors cores

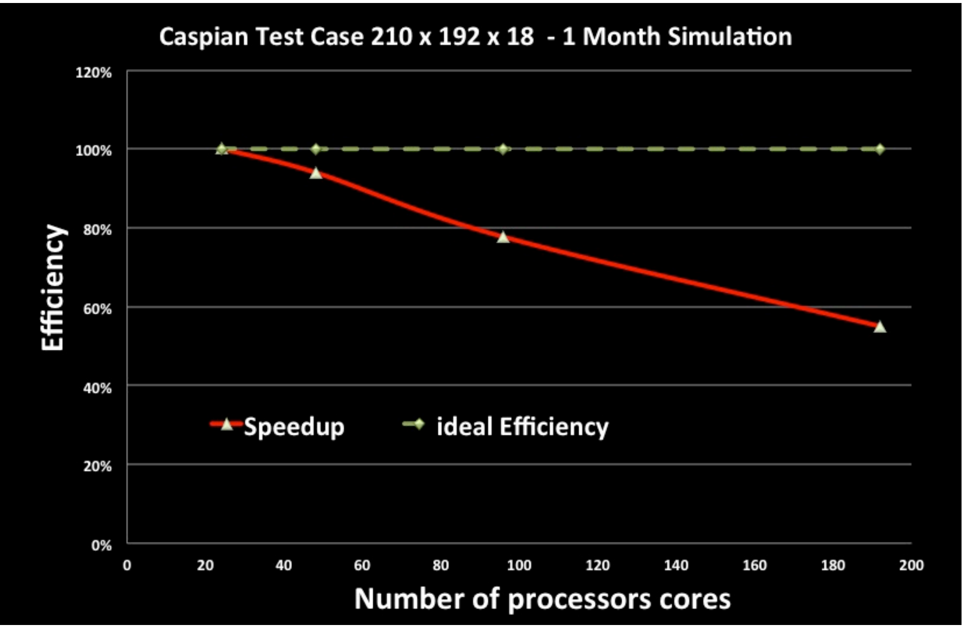
Efficiency

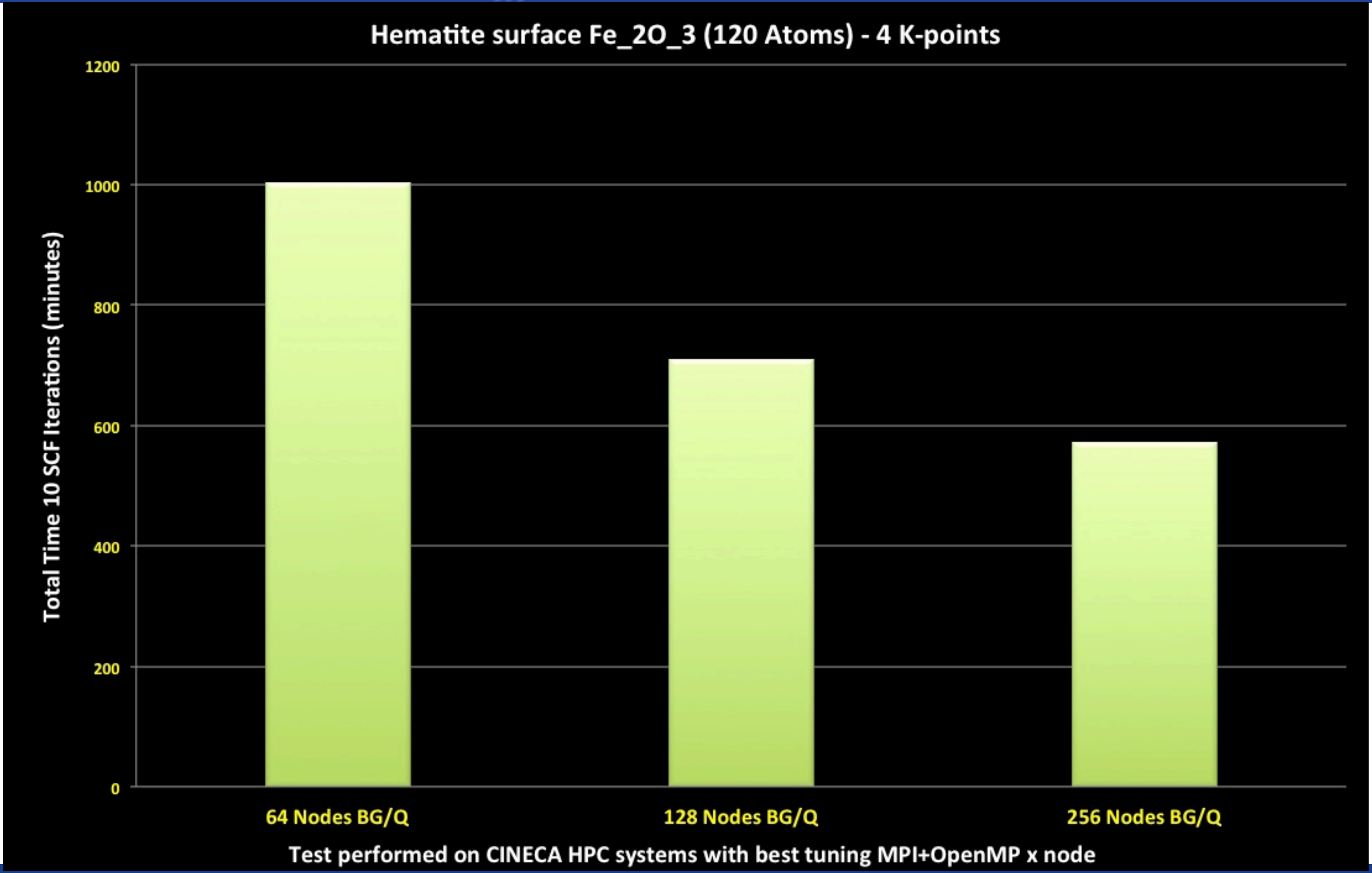
- Only embarrassing parallel algorithm can obtain an ideal Speedup
- The **Efficiency** is a measure of the fraction of time for which a processing element is usefully employed:

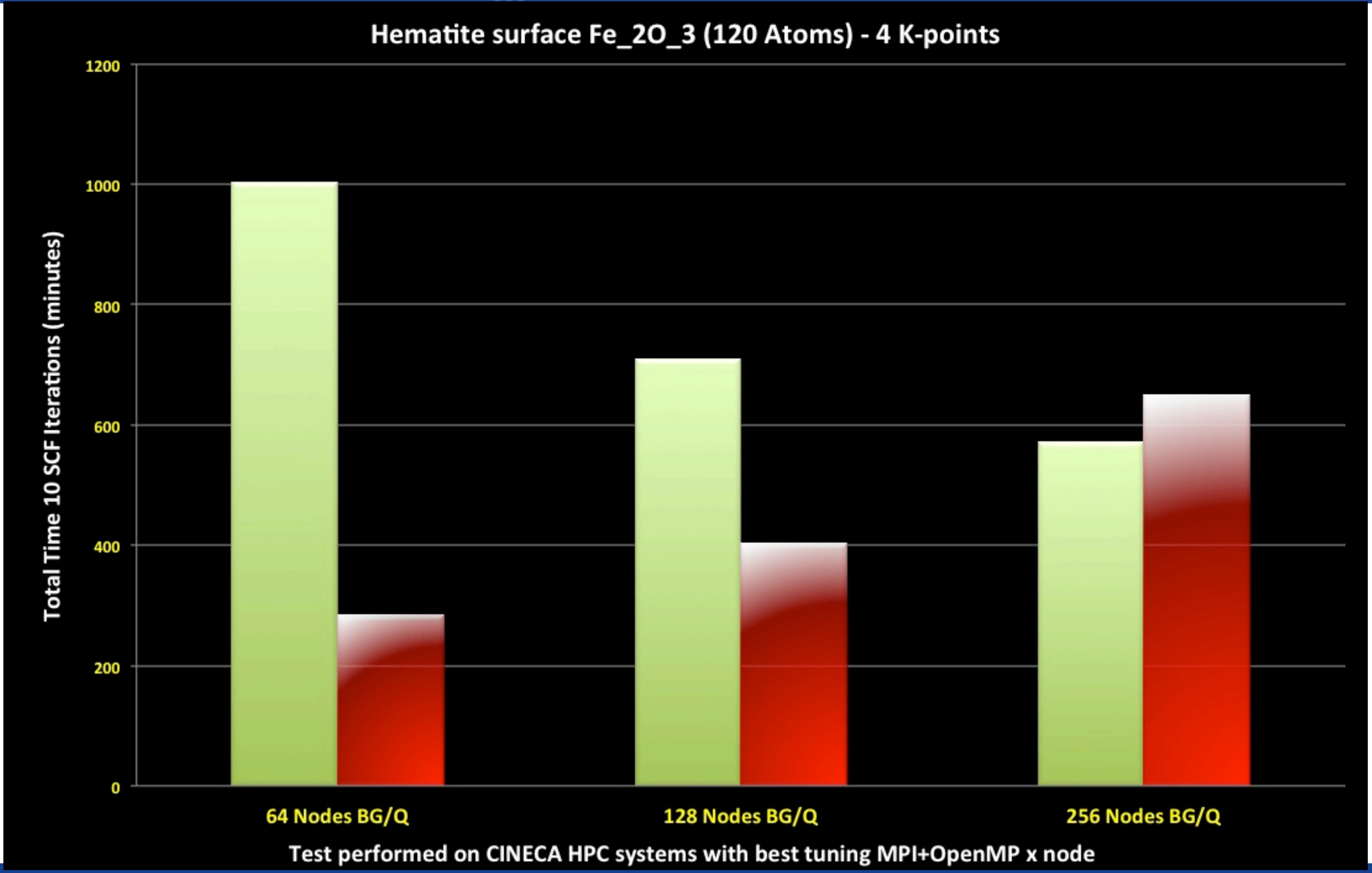
$$E_P = \frac{S_P}{p}$$

Efficiency


Caspian Test Case 210 x 192 x 18 - 1 Month Simulation
Sandy Bridge - new Argo's Intel Six-Core Processors
Total time of execution (seconds)
Number of processors cores
0
1000
2000
3000
4000
5000
6000
7000
24
48
96
192


Caspian Test Case 210 x 192 x 18 - 1 Month Simulation
Efficiency
Speedup
ideal Efficiency
Number of processors cores
0%
20%
40%
60%
80%
100%
120%
0
20
40
60
80
100
120
140
160
180
200

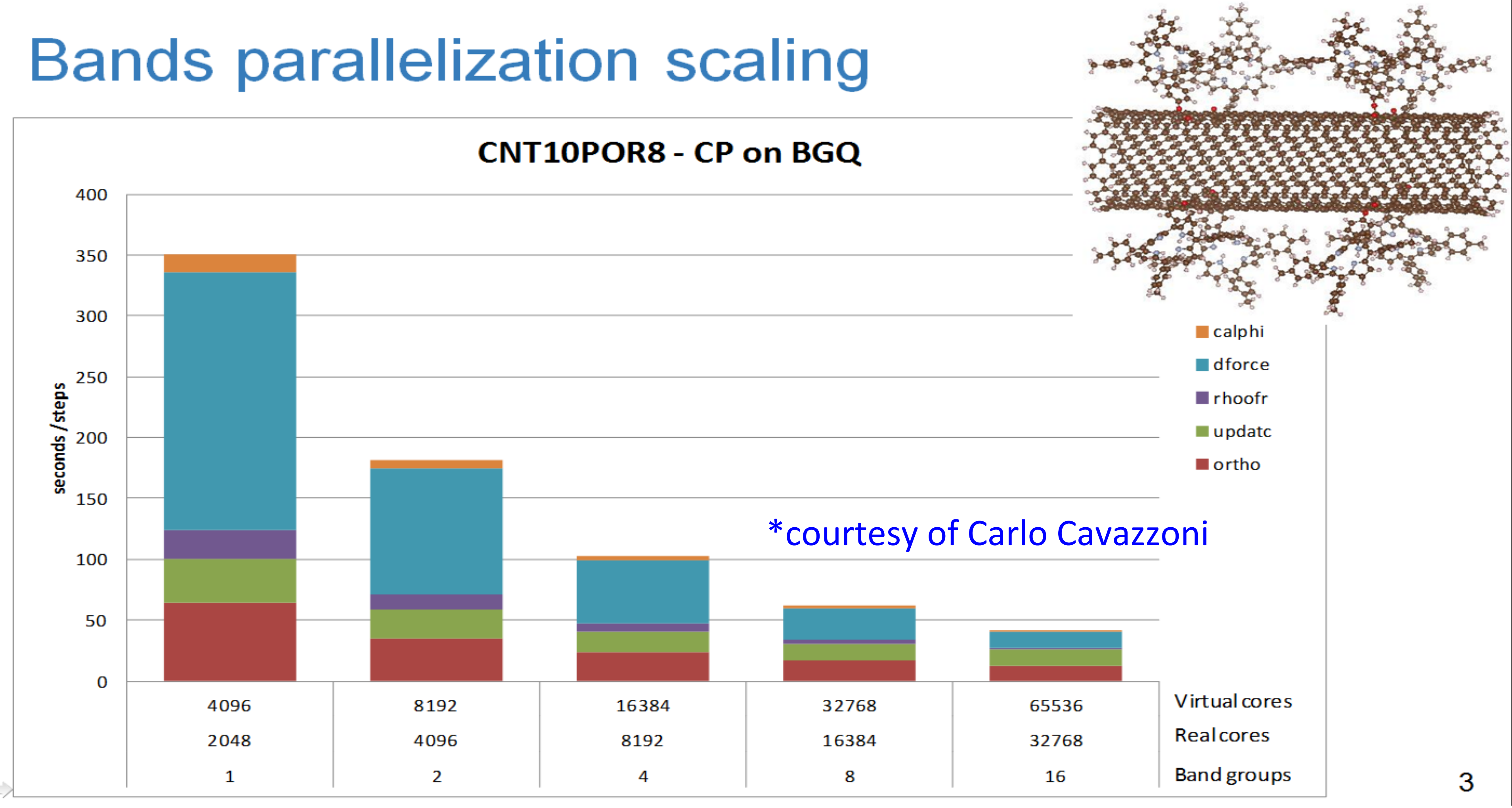

Hematite surface Fe_2O_3 (120 Atoms) - 4 K-points
Total Time 10 SCF Iterations (minutes)
1200
1000
800
600
400
200
0
64 Nodes BG/Q
128 Nodes BG/Q
256 Nodes BG/Q
Test performed on CINECA HPC systems with best tuning MPI+OpenMP x node


Hematite surface Fe_2O_3 (120 Atoms) - 4 K-points
Total Time 10 SCF Iterations (minutes)
1200
1000
800
600
400
200
0
64 Nodes BG/Q
128 Nodes BG/Q
256 Nodes BG/Q
Test performed on CINECA HPC systems with best tuning MPI+OpenMP x node

Scalability

- When we want consider the scalability of our problem we are interested in two main features:
  - how much faster do we go increasing the number of processes for a fixed problem size (strong scaling)
  - how does the application behave if we increase the problem size keeping the workload fixed per processors?

Bands parallelization scaling



How do we get the profiling? /1

- Code instrumentation

```
double cclock()
/*
     Return the second elapsed
     since Epoch (00:00:00 UTC, January 1, 1970)
*/

{
    struct timeval tmp;
    double sec;
    gettimeofday( &tmp, (struct timezone *)0 );
    sec = tmp.tv_sec + ((double)tmp.tv_usec)/1000000.0;
    return sec;

}
```

- Profiling tools

Tuning and Analysis Utilities

The GNU Profiler (GPROF)

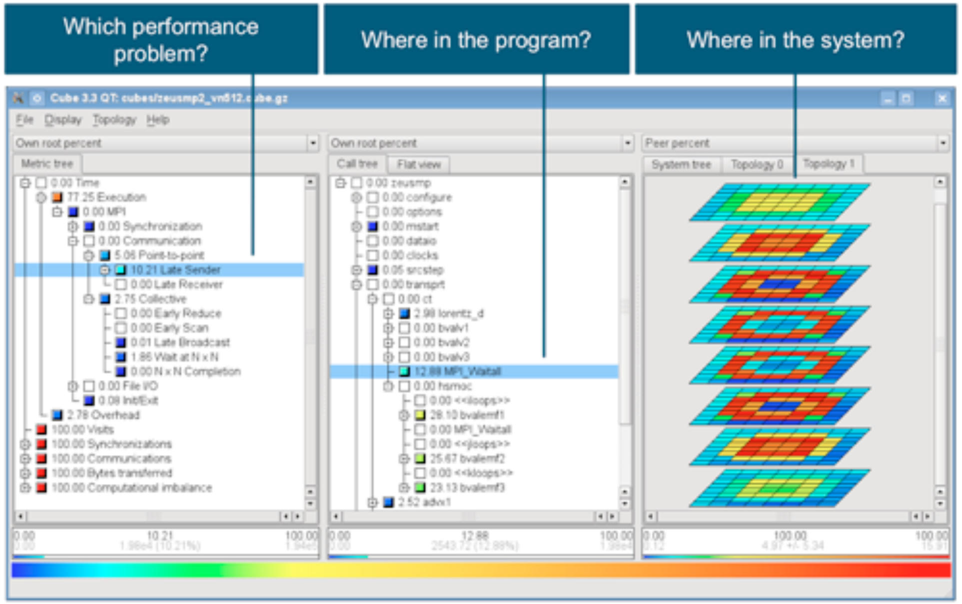
scalasca

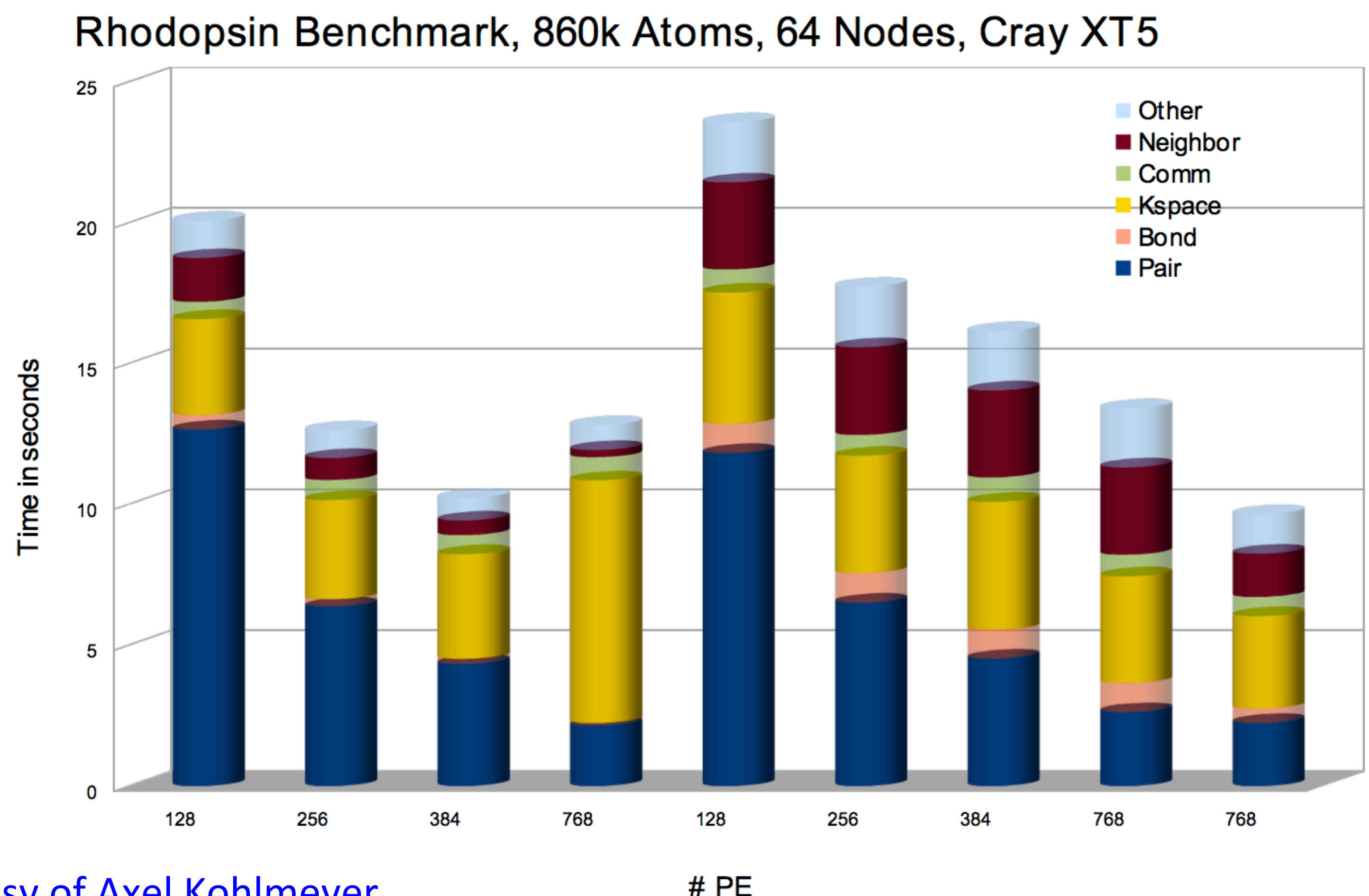
How do we get the profiling? /2

- Code instrumentation

```
Writing output data file c8_atm213_k111.save

init_run     :    91.65s CPU     91.65s WALL (       1 calls)
electrons    :  3366.51s CPU   3366.51s WALL (       1 calls)
forces       :    16.68s CPU     16.68s WALL (       1 calls)
stress       :   209.17s CPU    209.17s WALL (       1 calls)

Called by init_run:
wfcinit      :    68.98s CPU     68.98s WALL (       1 calls)
potinit      :     4.75s CPU      4.75s WALL (       1 calls)

Called by electrons:
c_bands      :  3000.94s CPU   3000.94s WALL (      23 calls)
sum_band     :   192.26s CPU    192.26s WALL (      23 calls)
v_of_rho     :     4.41s CPU      4.41s WALL (      24 calls)
mix_rho      :     6.72s CPU      6.72s WALL (      23 calls)

Called by c_bands:
init_us_2    :     2.12s CPU      2.12s WALL (      47 calls)
cegterg      :  2994.88s CPU   2994.88s WALL (      23 calls)

Called by *egterg:
h_psi        :   940.26s CPU    940.26s WALL (      70 calls)
g_psi        :    30.53s CPU     30.53s WALL (      46 calls)
cdiaghg      :  1223.83s CPU   1223.83s WALL (      69 calls)

Called by h_psi:
add_vuspsi   :    78.78s CPU     78.78s WALL (      70 calls)

General routines
calbec       :    65.14s CPU     65.14s WALL (      72 calls)
fft          :     9.65s CPU      9.65s WALL (     271 calls)
ffts         :     2.55s CPU      2.55s WALL (     474 calls)
fftw         :   894.47s CPU    894.51s WALL (   75284 calls)
davcio       :    32.45s CPU     32.45s WALL (      23 calls)

Parallel routines
fft_scatter  :   284.51s CPU    284.65s WALL (   76029 calls)
ALLTOALL     :    61.81s CPU     61.82s WALL (   75272 calls)
EXX routines

PWSCF        :      1h 1m CPU        1h 1m WALL
```

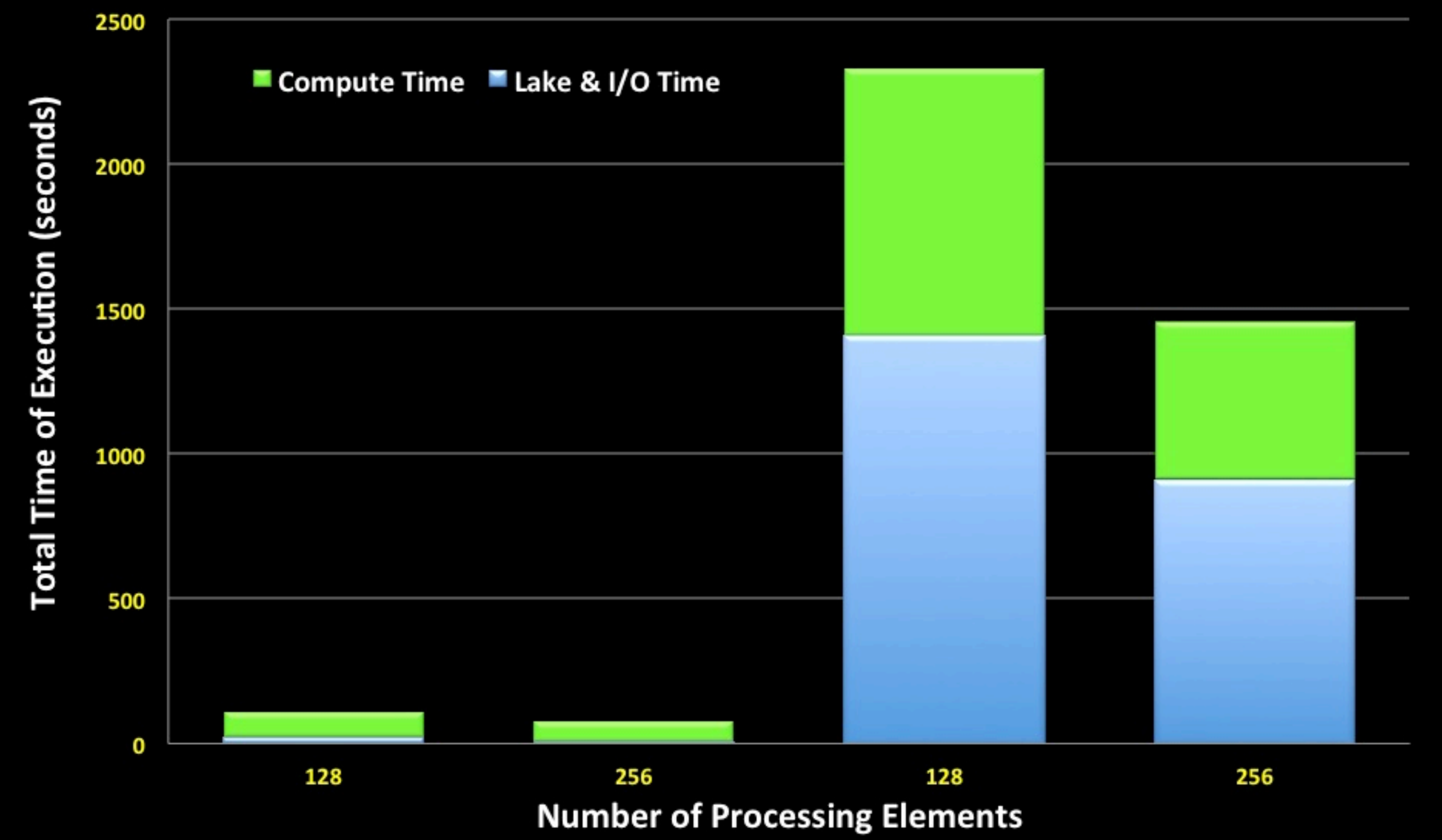
- Profiling tools





*courtesy of Axel Kohlmeyer

RegCM Code - Time of execution ARGO Vs. BG/Q



Conclusions

- How many processors shall I use?
- The performance analysis of a parallel program is needed to quantify the improvement we have with the parallelization
- Performance and scalability are usually requested for proposals to obtain access at large scale facility and significant amount of CPU-h
- Efficiency must be considered to quantify the cost of execution

Thanks for your attention!!

References

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