



The Abdus Salam
International Centre
for Theoretical Physics



IAEA
International Atomic Energy Agency



QUANTUM ESPRESSO

The QE-GPU Package

Ivan Girotto – igirotto@ictp.it

Information & Communication Technology Section (ICTS)

International Centre for Theoretical Physics (ICTP)

... in collaboration with Filippo Spiga (Cambridge University)



OUTLINE

- Technological Background
- Enabling of PWscf on platforms equipped with NVIDIA GPU/s
- Performance Analysis
- Conclusions



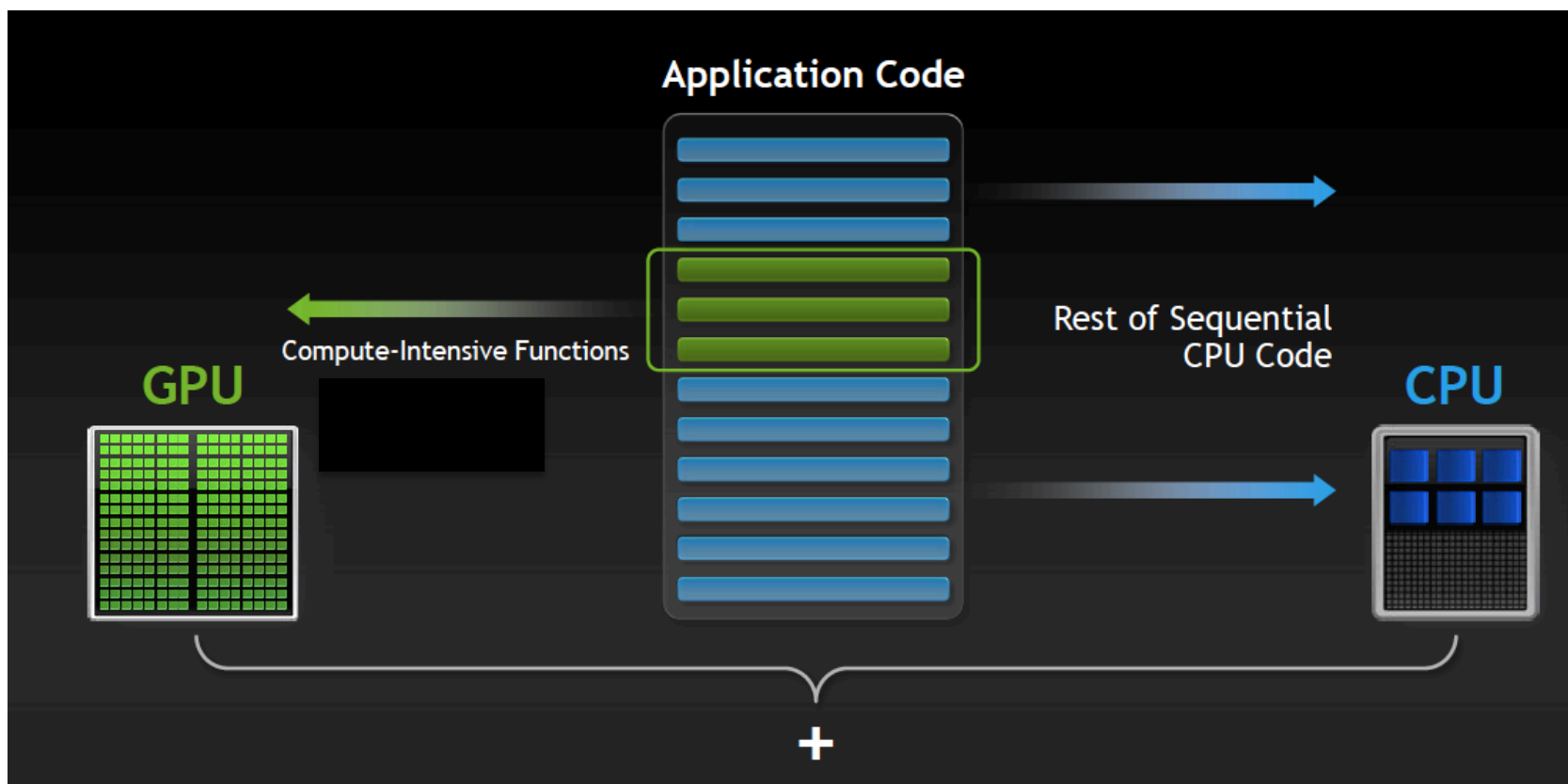
The Abdus Salam
**International Centre
for Theoretical Physics**

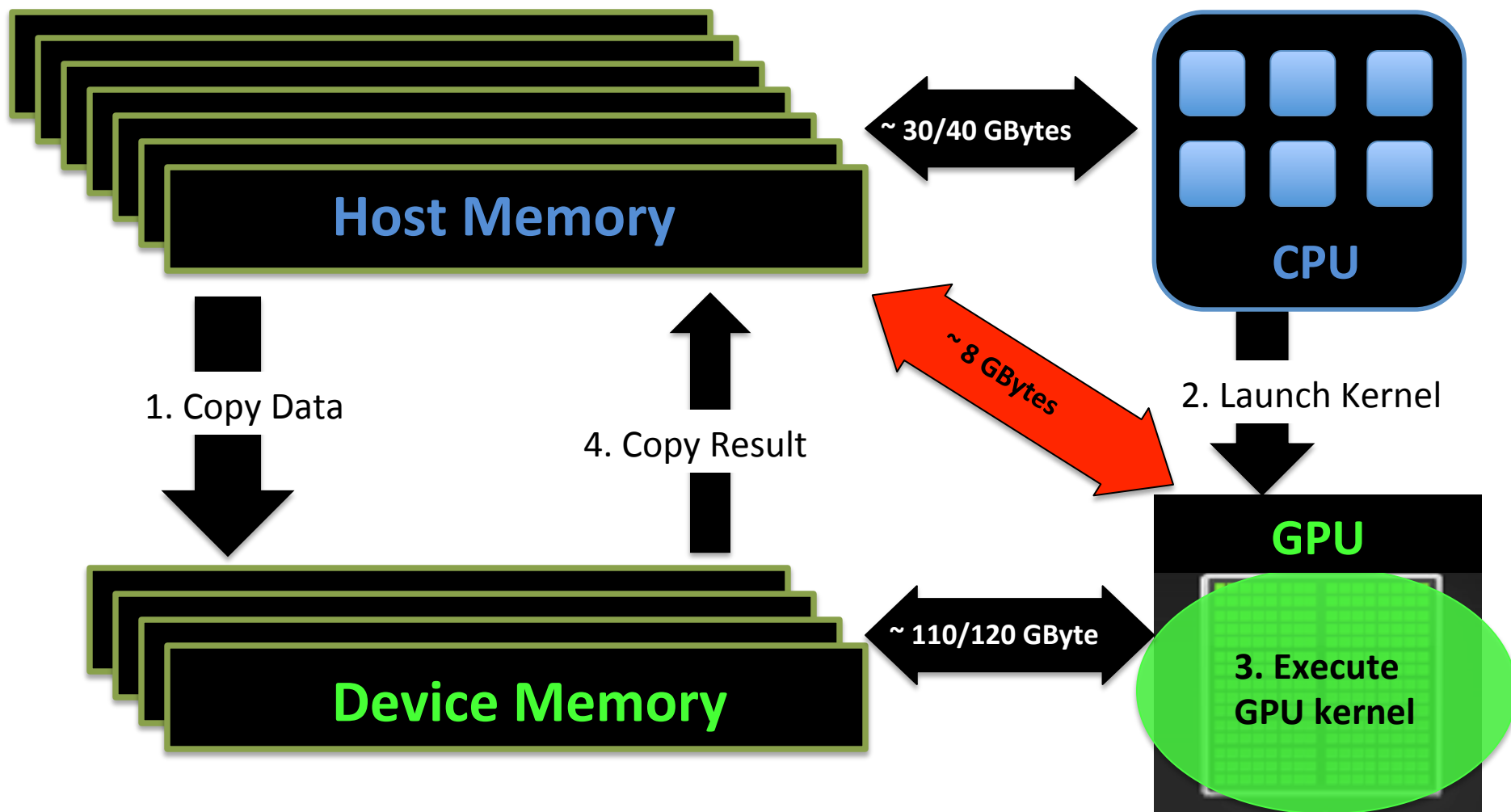


IAEA
International Atomic Energy Agency

TECHNOLOGICAL BACKGROUND

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?
- The DP peak of performance is done as follows:
 - #operations per clock cycle x frequency x #cores
 - we automatically reduce the DP power if we partially use the accelerator
- How much is my GPU better than my CPU?
- A relevant outcome is always a good balance of those issues



The Abdus Salam
**International Centre
for Theoretical Physics**



ENABLING OF PWSCF ON GPU

The Grounds

- The **almost** “perfect” storm
- PRACE-1IP and the Irish Experience
- Visibility
 - It is good having a business compliant product (grants, funding, PR 😊)
- Target: enabling of QE to high-end platforms

The Experienced Problems

- Low DP for consumer computing
- Huge effort for competitiveness and maintenance
- No portability even across generation of NVIDIA GPUs!!
- Software integration and validation



QE-GPU

Home » Projects » QE-GPU » Home

Summary

- >> Reporting
- >> Search
- >> Forums
- >> Tracker
- >> Docs
- >> News
- >> Files
- >> Lists
- >> SVN



Quantum ESPRESSO is an integrated suite of Open-Source computer codes for electronic-structure calculations and materials modeling at the nanoscale. Aim of this project is to create a "plugin-like" component for the standard Quantum ESPRESSO package that allows to exploit the capabilities of [NVIDIA](#) GPU graphics cards in order to allow materials scientists to do better and fast science.

[GPU acceleration](#) is currently available for the Plane-Wave Self-Consistent Field (PWscf) code and the energy barriers and reaction pathways through the Nudged Elastic Band method (NEB) package.

This project follows the same spirit of open-source. Researchers and computational scientists active in the field of electronic-structure calculations with expertise GPGPU computing are encouraged to participate in the project by contributing their own codes or by implementing their own ideas.

The QE+GPU distribution can be downloaded [here](#) (QE project files' section)

How to build QE-GPU (for additional information please refer to README.GPU):

After checkout the entire QE repository from SVN or untar QE-GPU.tar.gz in your working directory..

```
$ cd GPU
$ ./configure --enable-parallel --enable-openmp --enable-cuda --with-gpu-arch=20 --with-cuda-dir=/opt/packages/cuda/4.0/cuda --enable-magma
$ cd ..
$ make -f Makefile.gpu pw-gpu
```

'pw-gpu.x' executable will be placed under 'bin/'.

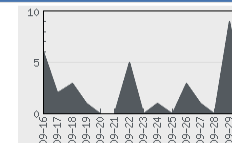
Do you want to try QE-GPU but you do not have (yet) GPU systems in your university or institution? Try [NVIDIA GPU Test Drive](#) !

NVIDIA along several hardware providers is offering [free](#) and exclusive test-drive to experience running your computational chemistry simulations faster with GPUs in dedicated small HPC systems and/or in cloud environments. The latest GPU-accelerated applications are pre-loaded so you do not need setup any hardware or software. Simply log on and run apps as usual, NO GPU PROGRAMMING EXPERIENCE IS REQUIRED.

Recent News

No news items found

Activity



Description

This independent repository collects various GPU piece of codes related to the Quantum ESPRESSO suite in order to exploit new hybrid CPU+GPU high performance computing systems.

Developer Info

Layla Martin-Samos
Ivan Girotto
nicola varini
Filippo Spina

Trove Categorization

- Development Status: 5 - Production/Stable
- Intended Audience: Developers
- License: GNU General Public License (GPL)
- Programming Language: C, Fortran, Python
- Topic: Density-Functional Theory calculation, Libraries, Quantum ESPRESSO related

Development strategy in a Nutshell

- Important compromise between performances and code re-factoring
- Scalability & Reliability are clear objectives (No trivial Init Module):
 - Multiple processes mapping on single GPU
 - Memory control and management
 - Data transfer overlapping (Pinned memory or not Pinned memory?)
 - Result consistency with the CPU version
- Massive focus on best exploitation of GPU libraries
- Few small routines developed in CUDA (< few hundreds lines)
- Extension to the NEB calculation

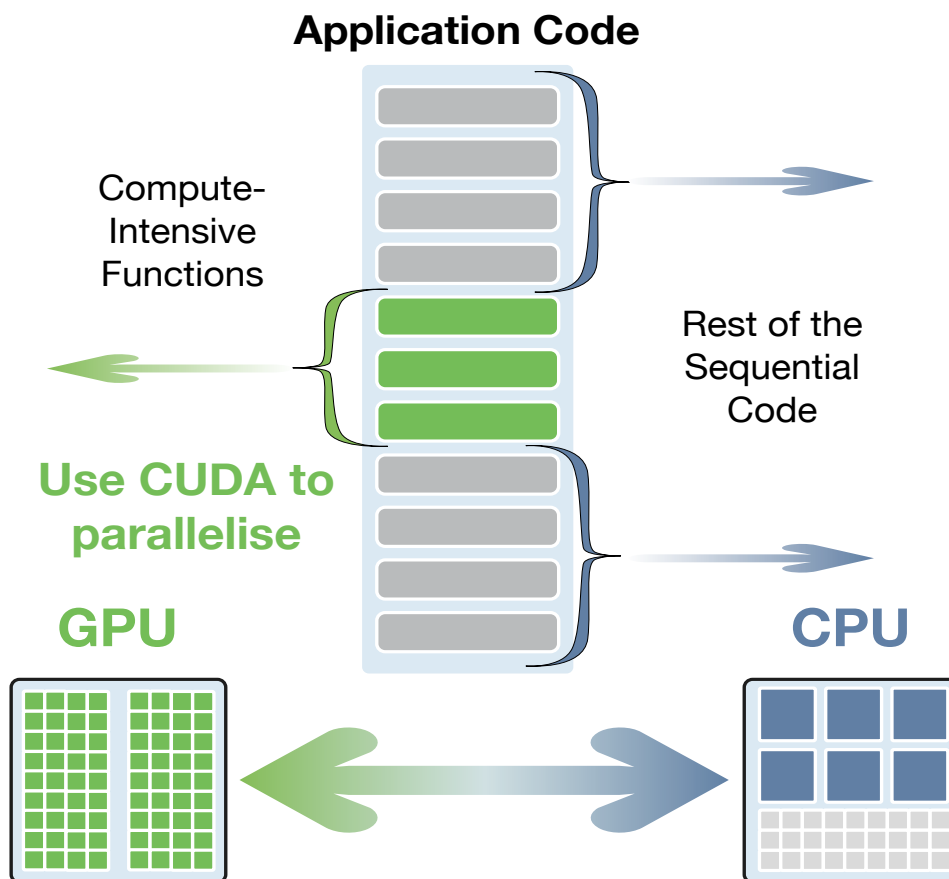
Speed-up Scientific Codes

➤ **Directives (OpenACC)**

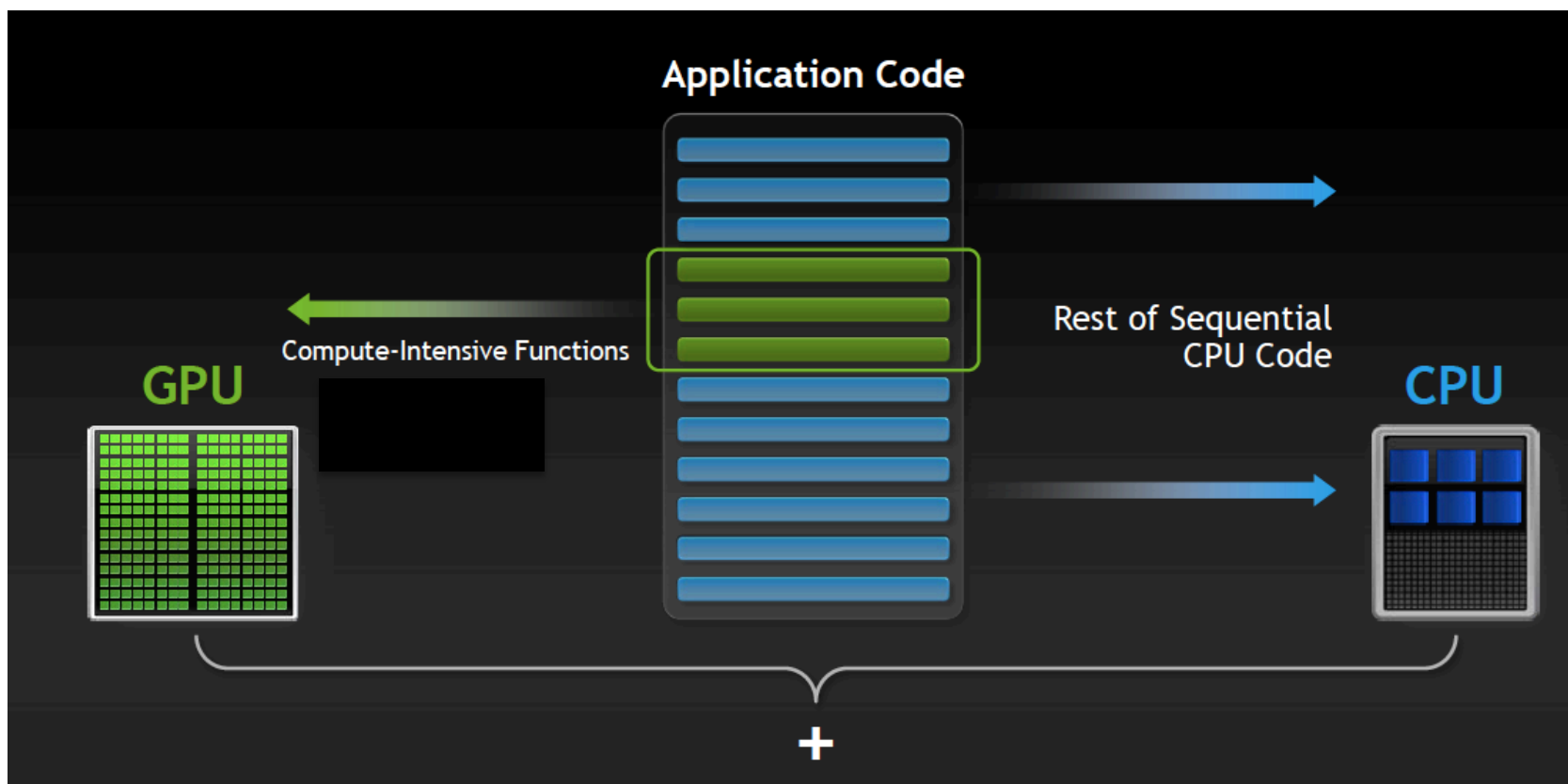
➤ **Libraries**

➤ **CUDA (or OpenCL)**

**3 Way
to Accelerate on GPU**



The General Concept of Accelerated Computing

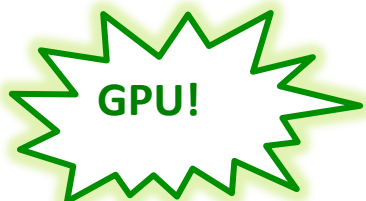


Computational Bottlenecks in QE

- Calculation of density, $n(\mathbf{r}) = \sum |\psi(\mathbf{r})|^2$ (+ augmentation terms for USPP): FFT + linear algebra (matrix-matrix multiplication)
- Calculation of potential, $V(\mathbf{r}) = V_{xc}[n(r)] + V_H[n(\mathbf{r})]$: FFT + operations on real-space grid
- Iterative diagonalization (SCF) / electronic force (CP) calculation, $H\psi$ products: FFT + linear algebra (matrix-matrix multiplication)
- Subspace diagonalization (SCF) / Iterative orthonormalization of Kohn-Sham states (CP): diagonalization of $N_e \times N_e$ matrices + matrix-matrix multiplication

*courtesy of Prof. Paolo Giannozzi

Levels of parallelism in QE



Images

- Only for Nudged Elastic Band (NEB) calculations

K-points

- Distribution over k-points (if more than one)
- Scalability over k-points (if more than one)
- No memory scaling

Plane-waves

- Distribution of wave-function coefficients
- Distribution of real-grid points
- Good memory scale, good overall scalability, LB

Linear algebra &
task groups

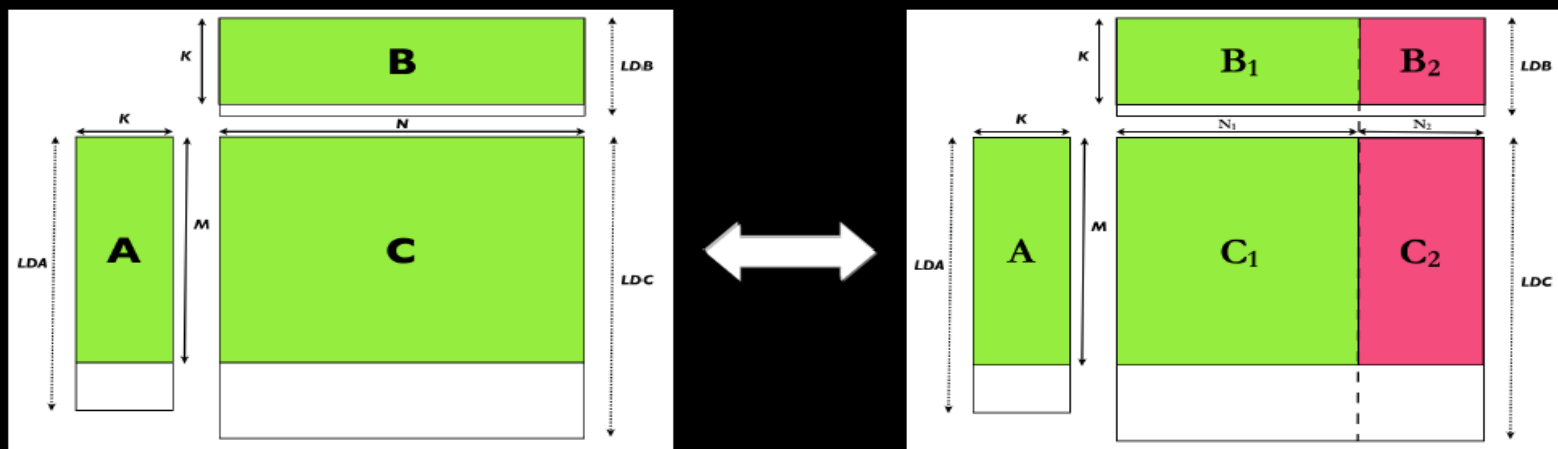
- Iterative diagonalization (fully-parallel or serial)
- Smart grouping of 3DFFT to reduce *compulsory* MPI communications

Multi-threaded
kernels

- OpenMP handled *explicitly* or *implicitly*
- Extend the scaling on multi-core machines with “limited” memory

Existing work, Dgemm for Linpack HPL

DGEMM: $C = \alpha A B + \beta C$



$$\text{DGEMM}(A,B,C) = \text{DGEMM}(A,B_1,C_1) \cup \text{DGEMM}(A,B_2,C_2)$$

The idea can be extended to multi-GPU configuration and to handle huge matrices

Find the optimal split, knowing the relative performances of the GPU and CPU cores on DGEMM

Phillips & Fatica http://www.nvidia.com/content/GTC-2010/pdfs/2057_GTC2010.pdf

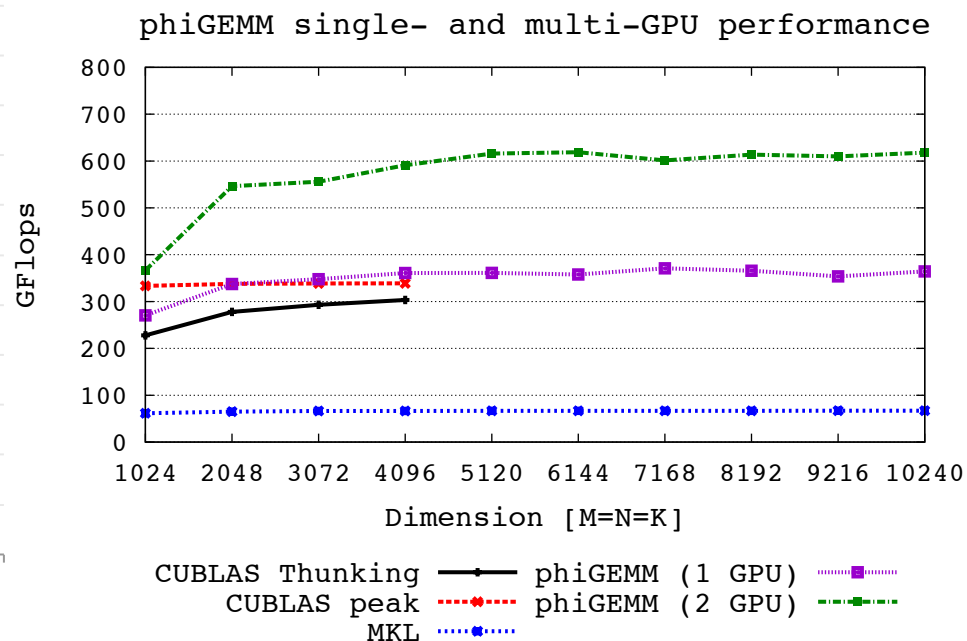
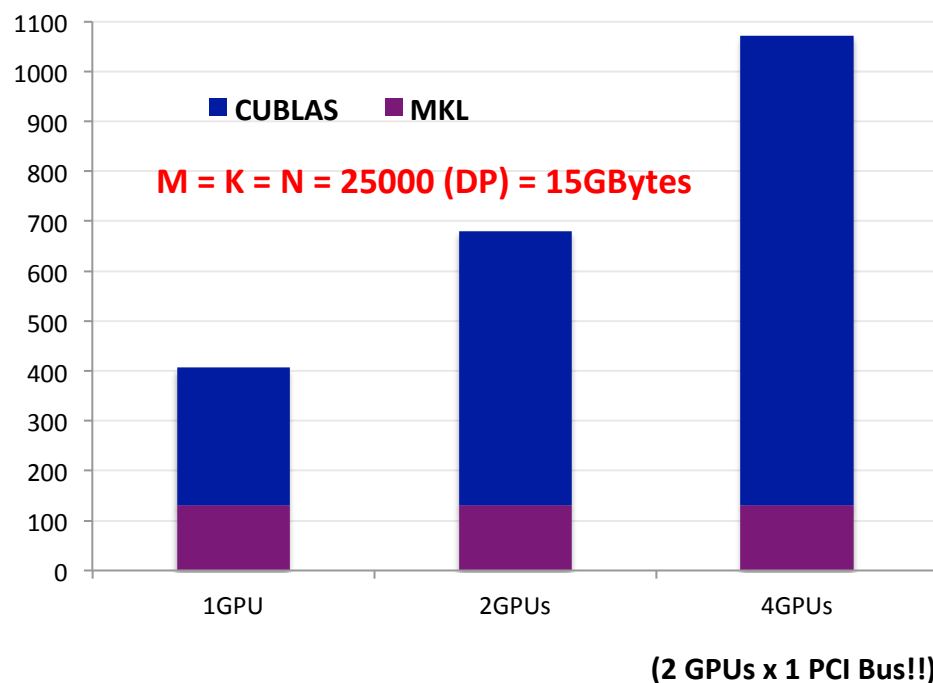
The starting point: Φ GEMM

- [*]GEMM implemented
- Transparent data transferring
- Recursive splitting for big matrixes (no limited by GPU Memory)
- Special-K strategy for rectangular matrixes
- Almost transparently linkable
- Possible profiling report for each [*]GEMM call
- Possible mappings CPU-processes:GPUs
 - 1:N => MultiGPU version
 - N:1 => 1+ MPI processes x single GPU



<http://qe-forge.org/gf/project/phigemmm/>

Φ GEMM



CUFFT

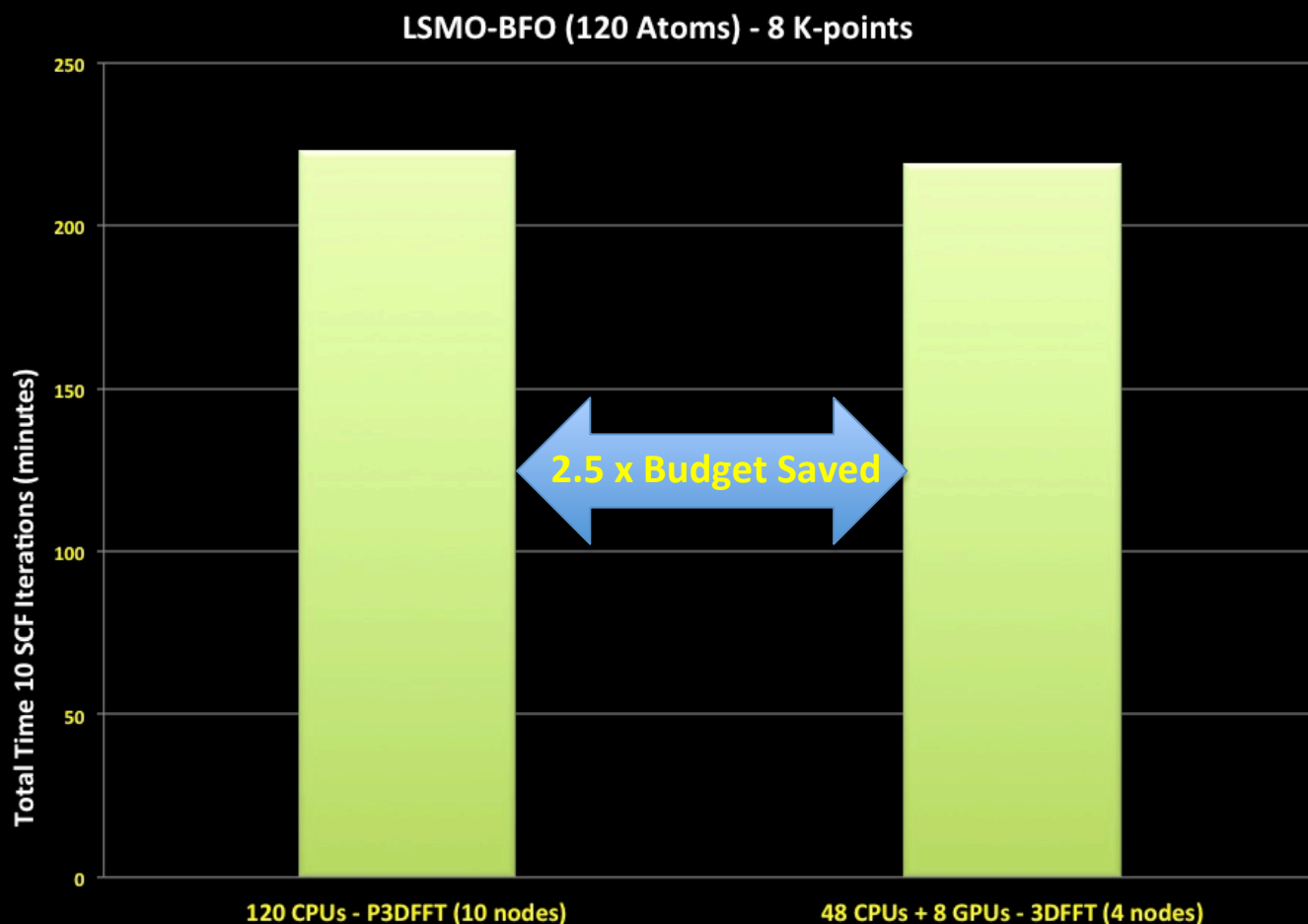
- NVIDIA library to interface call to FFTW
- Used only on serial version and if using `-D__USE_3D__FFT` (non distributed plane-waves)
- Good speedup Vs CPU for small number of MPI processes
- C code for wrapping CUFFT completely encapsulated in separated files
- 3DFFT on distributed data still under implementation

<http://developer.nvidia.com/cuda/cufft>

```

IF ( gamma_only ) THEN
  #if (defined(__CUDA) && ... )
    ierr = vloc_psi_cuda ( lda, dffts%nnr, dffts%nr1x, dffts%nr2x, dffts%nr3x, &
      n, m, psi, vrs(1,current_spin), hpsi, igk(1:), nls(1:), &
      nls(1:), ngms, ngm)
    !
  #else
    CALL vloc_psi_gamma ( lda, n, m, psi, vrs(1,current_spin), hpsi )
  #endif
ELSE IF ( noncolin ) THEN
  !
  CALL vloc_psi_nc ( lda, n, m, psi, vrs, hpsi )
ELSE
  #if (defined(__CUDA) && ... )
    ierr = vloc_psi_cuda_k ( lda, dffts%nnr, dffts%nr1x, dffts%nr2x, dffts%nr3x, &
      n, m, psi, vrs(1,current_spin), hpsi, igk(1:), nls(1:), ngms)
  #else
    CALL vloc_psi_k ( lda, n, m, psi, vrs(1,current_spin), hpsi )
  #endif
END IF

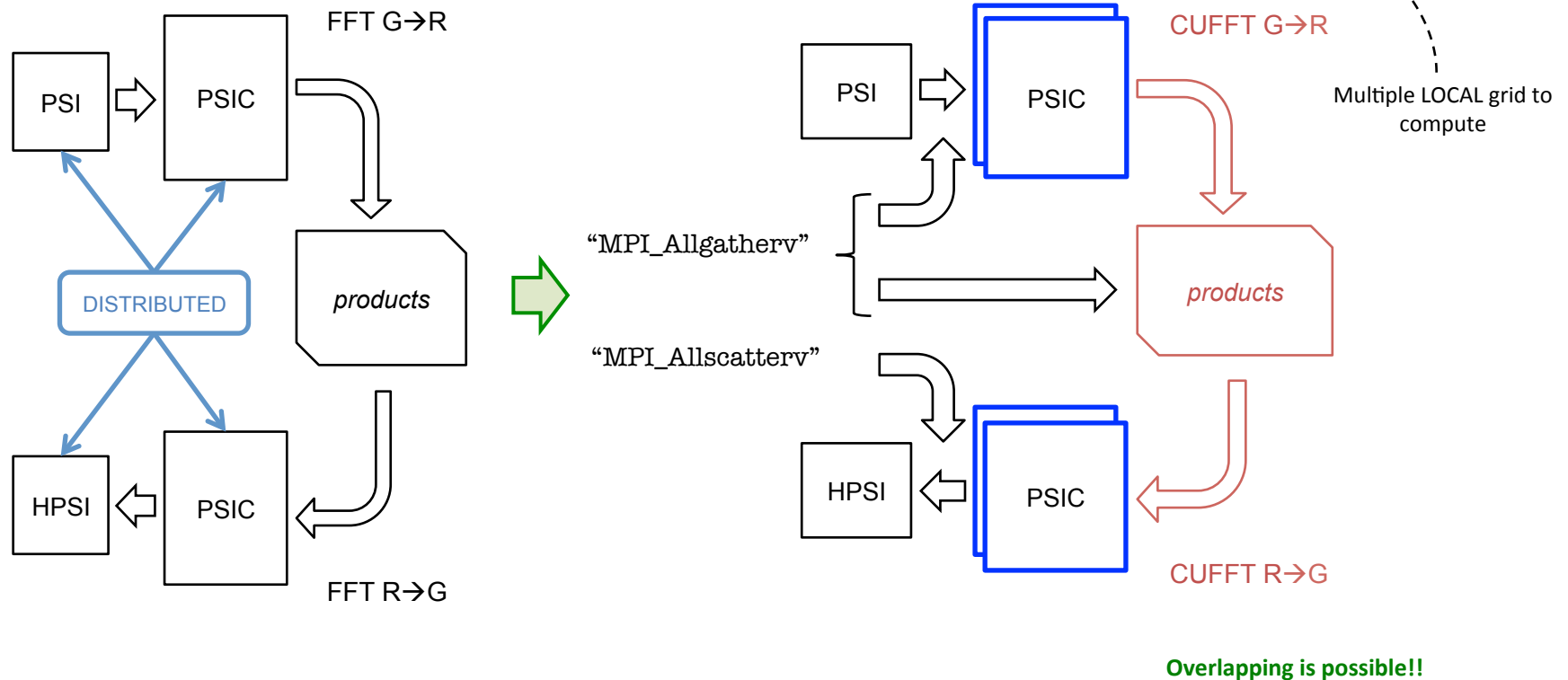
```



Test performed on CINECA PLX GPU Cluster with best tuning MPI + OpenMP x node
(274 x iDataPlex DX360M2, dual 6-cores Intel Xeon X5550 2.66 GHz + 2 NVIDIA M2070)

Performance Results /1

Parallel vLOC_PSI



MAGMA

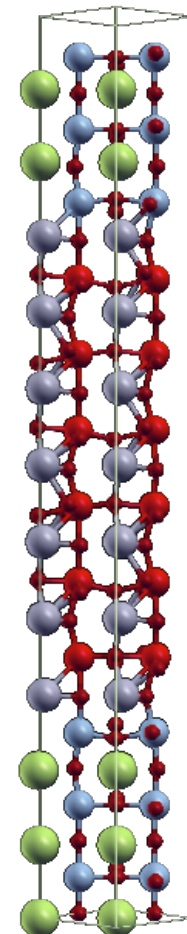
- LAPACK library for GPU systems

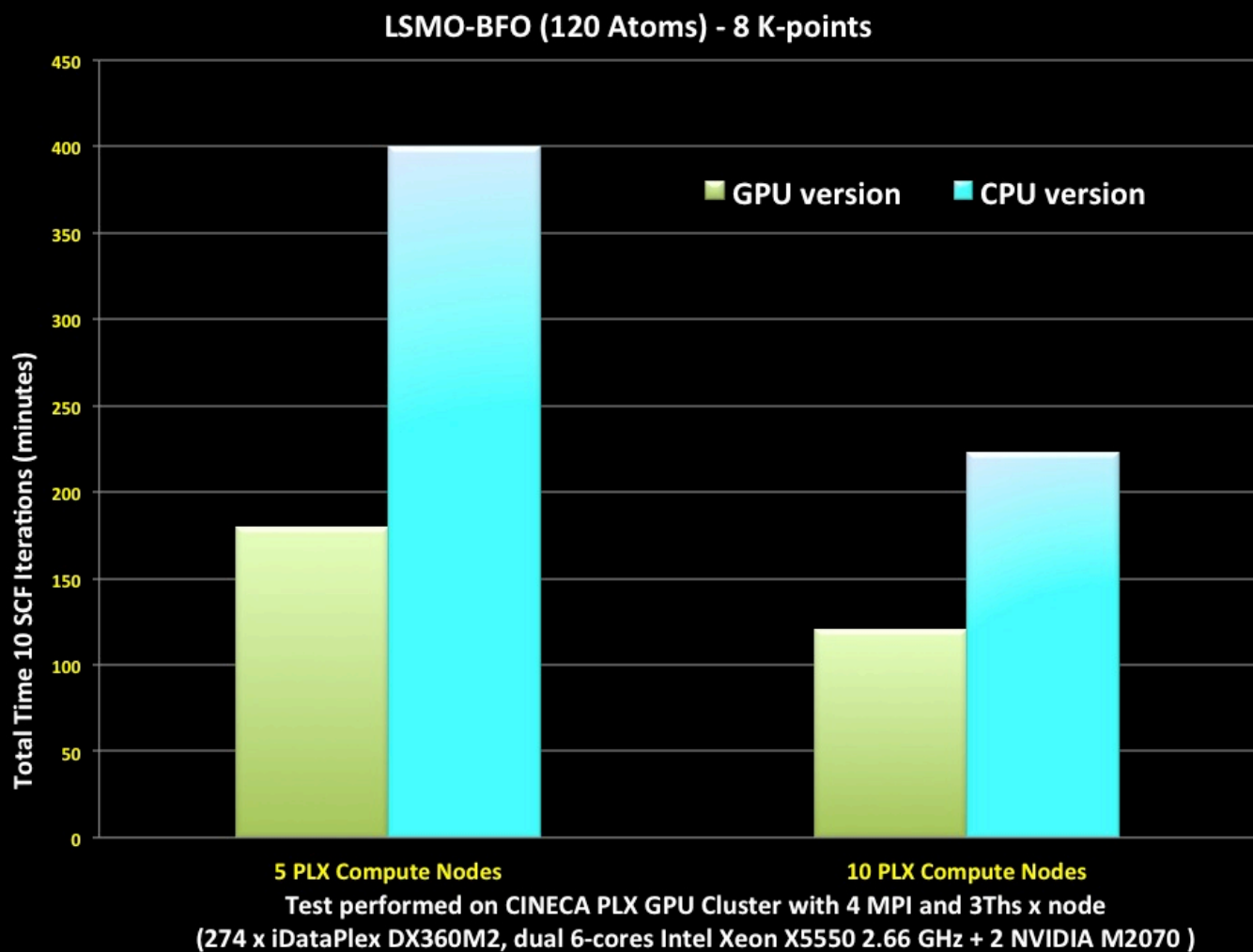
```
#if defined(__CUDA) && defined(__MAGMA)
    CALL start_clock( 'MAGMA_DSYGVD' )
    CALL magmaf_dsygvd( 1, 'V', 'U', n, v, ldh, s, &
                        ldh, e, work, lwork, iwork, liwork, info )
    CALL stop_clock( 'MAGMA_DSYGVD' )
#else
    CALL start_clock( 'DSYGV' )
    CALL DSYGV( 1, 'V', 'U', n, v, ldh, s, ldh, e, &
               work, lwork, info )
    CALL stop_clock( 'DSYGV' )
#endif
```

<http://icl.cs.utk.edu/magma/>

Best Practice /1

- Scientific case: $\text{La}_{2/3}\text{Sr}_{1/3}\text{MnO}_3$ / BiFeO_3 magnetic heterostructure (120 atoms)
- PI: Rodrigo Neumann Barros Ferreira, PhD Student Solid State Physics Department – Physics Institute, Rio de Janeiro Federal University
- Description: 1024 electrons, 615 different quantum-mechanical states considered, **8 k-points** for the integration over the Brillouin zone.
- Goal: exploit QE *pool* parallelism and GPU → keep the FFT local by using `npool=npocs`

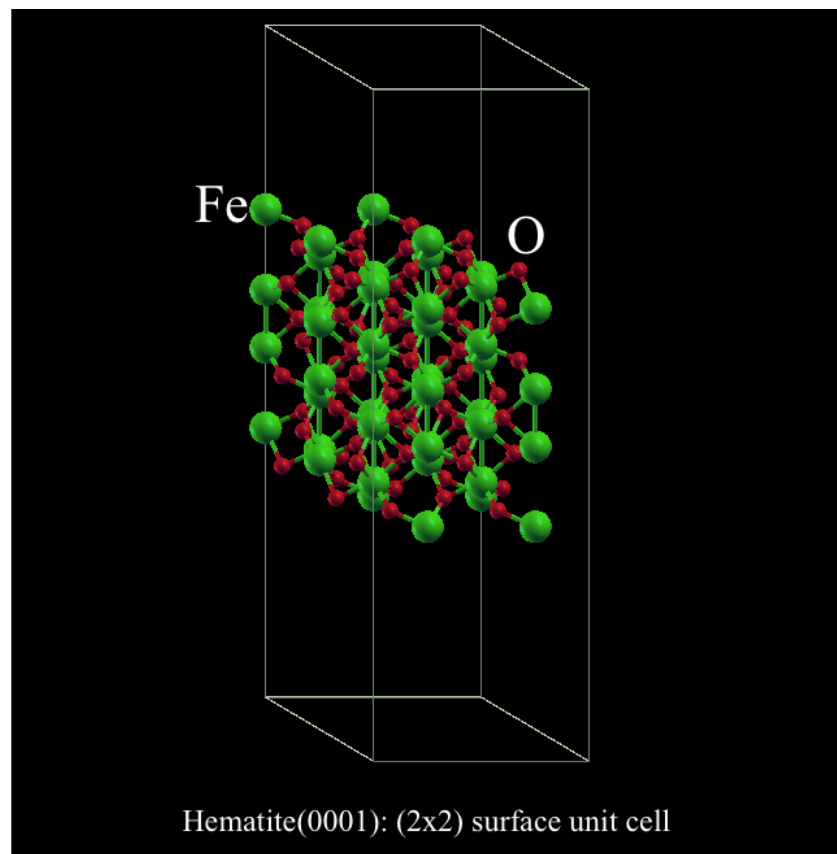




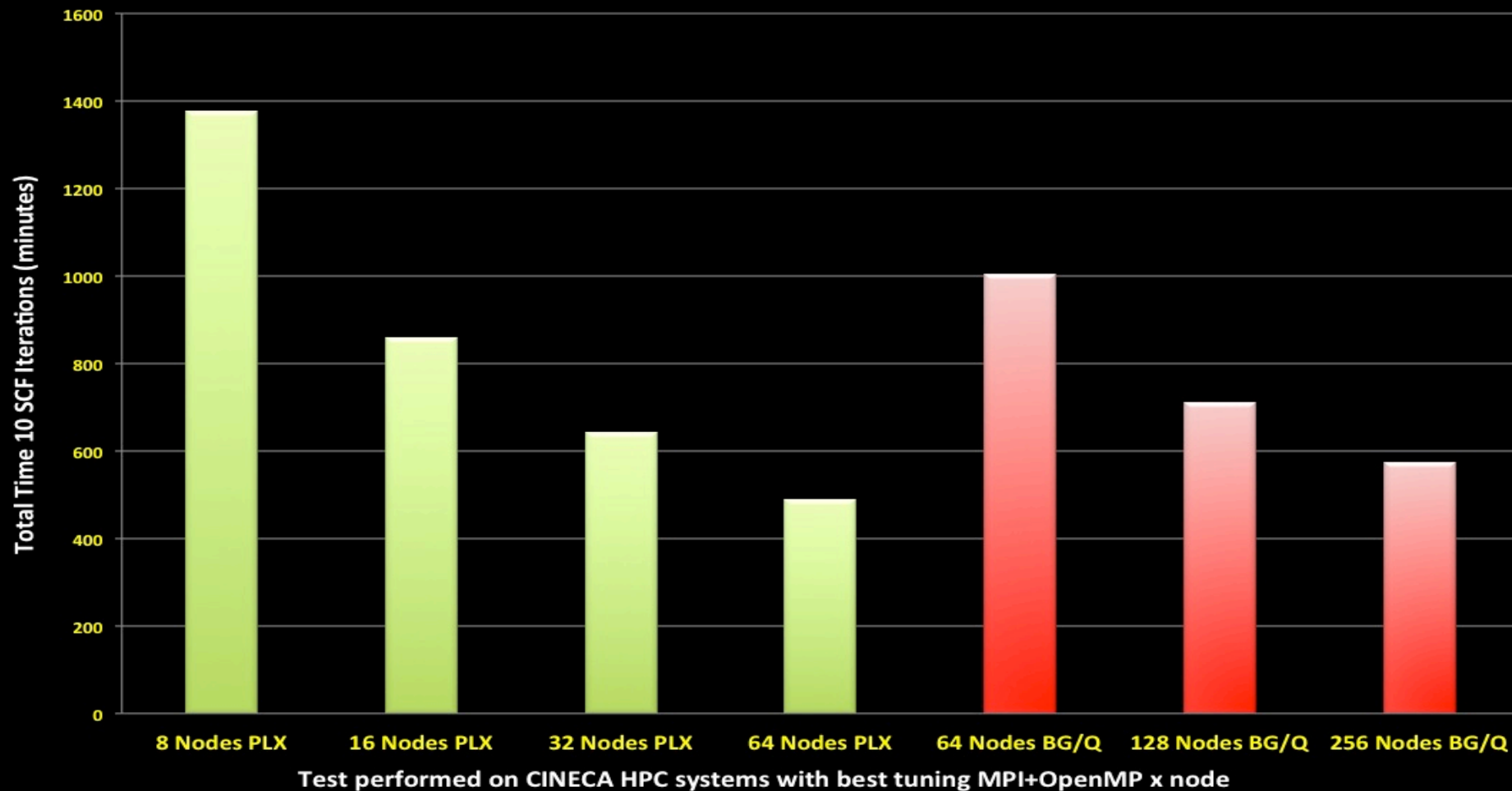
Performance Results /2

Best Practice /2

- Scientific case:
Fe₂O₃ (120 atoms)
- PI: Dr. Manh Thuong Nguyen,
Post Doctoral Fellow, The
Abdus Salam (ICTP)
- Description: Hematite surface,
1200 electrons, **4 k-points**
- Goal: exploit PWscf on both
GPU and IBM BG/Q systems



Hematite surface Fe₂O₃ (120 Atoms) - 4 K-points



State of The Art

- If compared with High-End multi-cores platforms, the GPU porting impacts when the communication saturates the MPI traffic and it is inescapable to reduce the number of processes per node
- Better tuning should be done to exploit the High-Throughput model. Huge number of independent systems at the same time
- Few references below:

<https://hpcforge.org/plugins/mediawiki/wiki/pracewp8/images/4/40/MarzariPRACE.pdf>

The high-throughput highway to computational materials design, Stefano Curtarolo, Gus L. W. Hart, Marco Buongiorno Nardelli, Natalio Mingo, Stefano Sanvito & Ohad Levy, Nature Materials 12, 191–201 (2013) doi:10.1038/nmat3568

Conclusion

- The porting of legacy code is not impossible. But it is a considerable effort.
- USE the GPU PWscf where NVIDIA GPUs are available!!
- The phase of tuning shouldn't scare, it is needed on all the High-End systems!
- Multithreading is inescapable to best exploit the CPU platform

Further development

- Code enabling on NVIDIA GPUs of current generation K20
- Feasibility study for the CP code and EXX section of the PWscf code
- Porting on other accelerated platforms (i.e., Intel-MIC?)



Acknowledgements:

- Filippo Spiga (Cambridge University)
- Paolo Giannozzi (Udine University)
- Carlo Cavazzoni (CINECA)
- Layla Martin-Samos (University of Nova Gorica)
- Rodrigo Neumann Barros Ferreira (Rio de Janeiro Federal University)
- Manh Thuong Nguyen (ICTP)



References

- P. Giannozzi and et al. QUANTUM ESPRESSO: a modular and open-source software project for quantum simulations of materials. Journal of Physics: Condensed Matter, 21(39), 2009.
- F. Spiga and I. Girotto, "phiGEMM: a CPU-GPU library for porting Quantum ESPRESSO on hybrid systems", Proceeding of 20th Euromicro International Conference on Parallel, Distributed and Network-Based Computing (PDP2012), Special Session on GPU Computing and Hybrid Computing, IEEE Computer Society, (ISBN 978-0-7695-4633-9), pp. 368-375 (2012)
- M. Fatica, "Accelerating LINPACK with CUDA on heterogeneous clusters." GPGPU-2: Proceedings of 2nd Workshop on General Purpose Processing on Graphics Processing Units (New York, NY, USA), ACM, 2009, pp. 46--51.
- Rob Farber, CUDA Application Design and Development, Morgan Kaufmann; 1 edition (November 14, 2011), ISBN-13: 978-0123884268