Use of QE in HPC: overview of implementation and usage of the QE-GPU

Ivan Girotto – igirotto@ictp.it

Information & Communication Technology Section (ICTS)

International Centre for Theoretical Physics (ICTP)

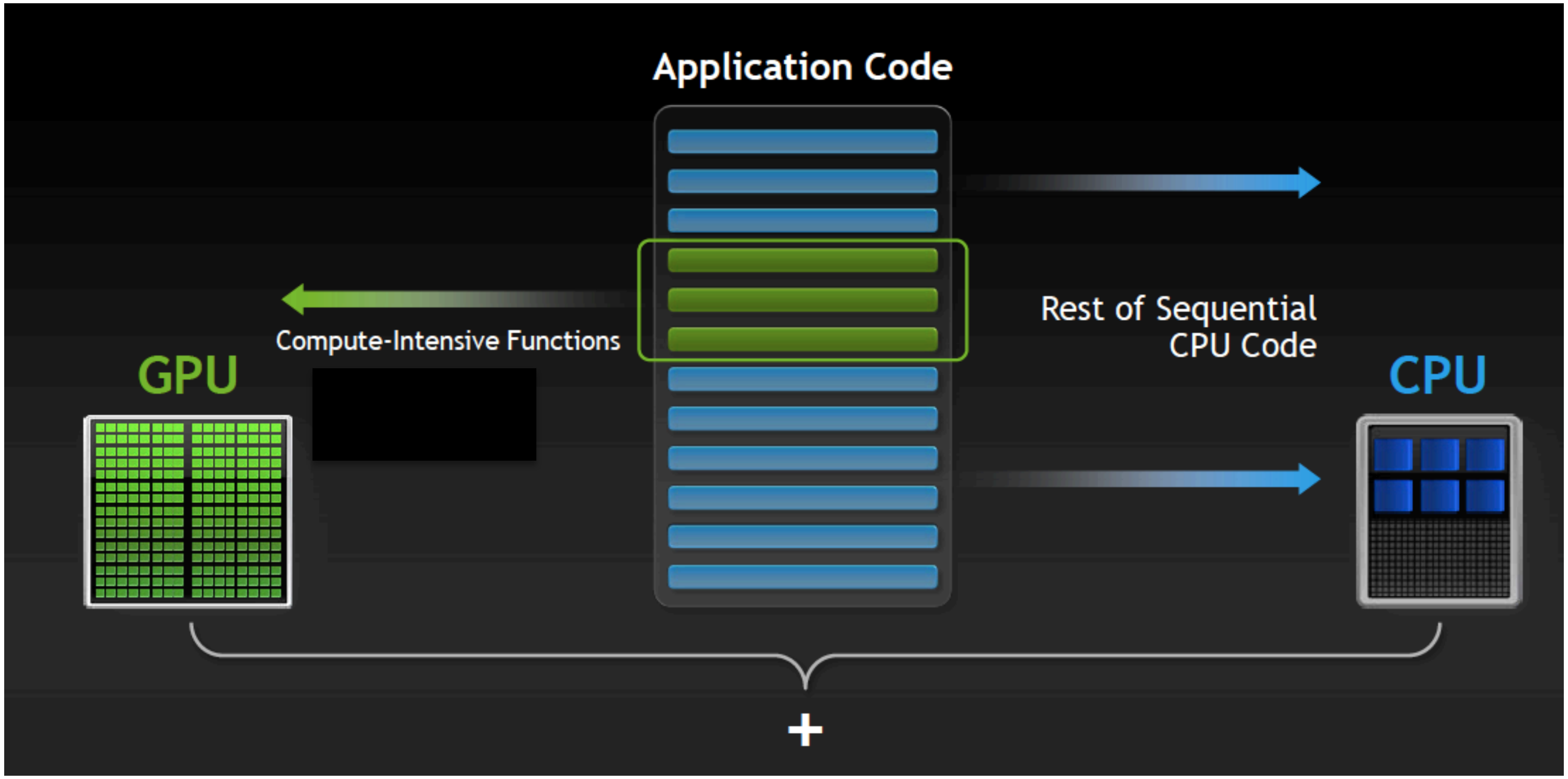
... in collaboration with Filippo Spiga (Cambridge University)

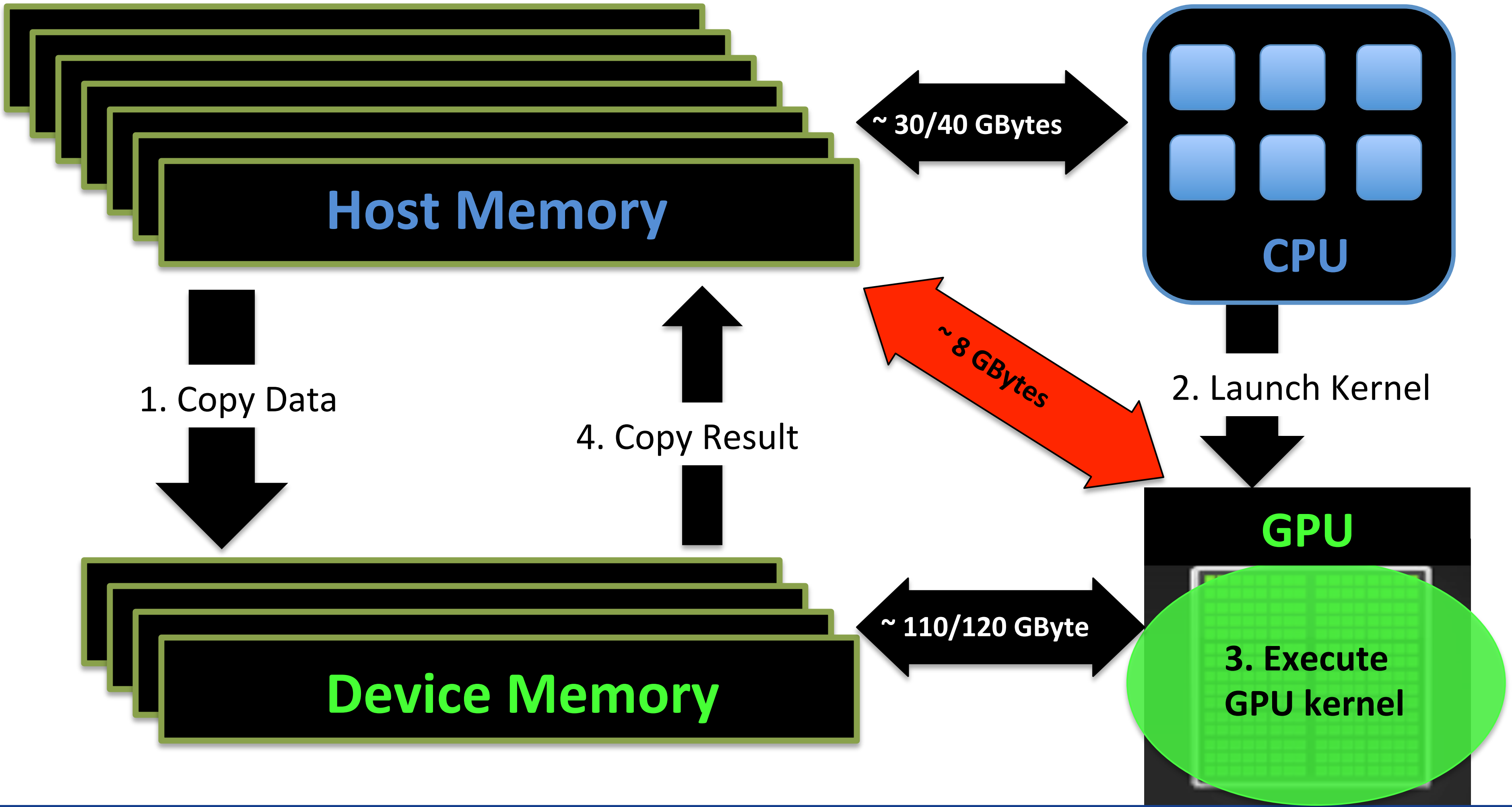
OUTLINE

- Technological Background
- Development foundation
- Usage
- Performance Analysis
- Conclusions

TECHNOLOGICAL BACKGROUND

The General Concept of Accelerated Computing




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

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

ENABLING OF PWSCF ON GPU

The Grounds

- The almost "perfect" storm
- PRACE-1IP and the Irish Experience
- Visibility
  - having a business compliant product helps (grants, funding, PR ☺)
- Targets:
  - enabling of QE to high-end platforms
  - provide an additional feature to the package
  - make it comparable with other codes enabled on GPU 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

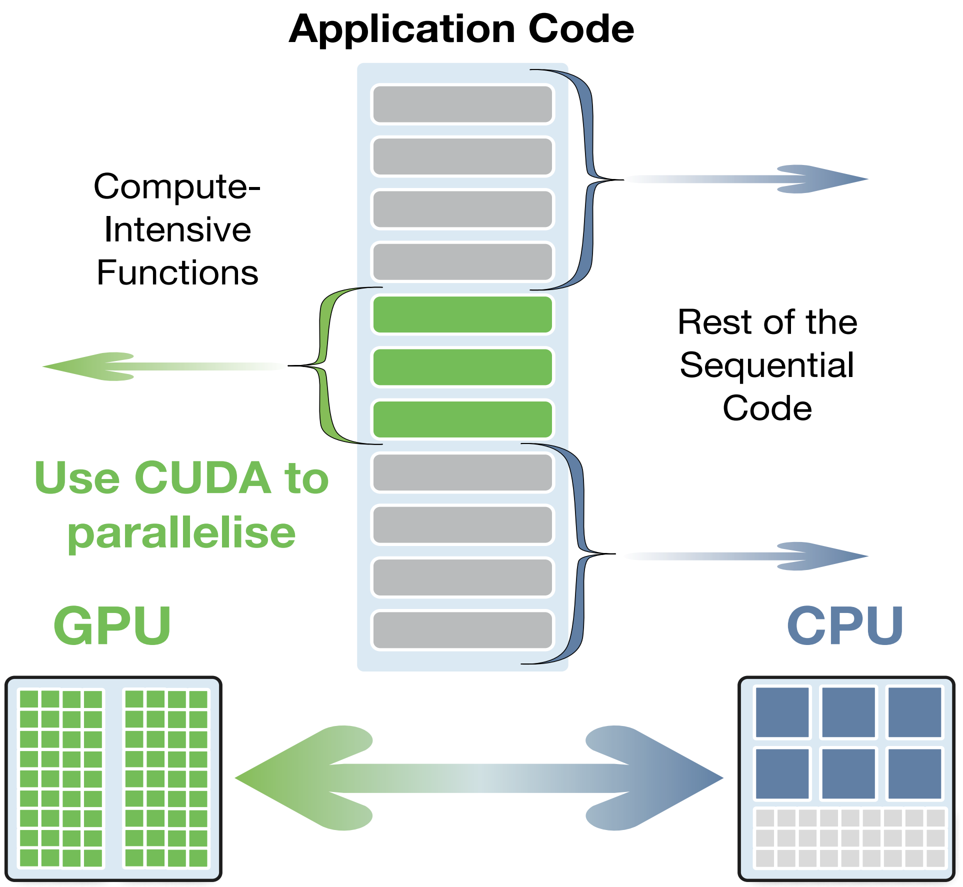
The QE-GPU repository

- The QE-GPU is now a plug-in maintained in a separated repository
- Please, visit https://github.com/fspiga/QE-GPU
- It must be downloaded a part and added into the QE $ROOT_DIRECTORY
- We always look for volunteers!!

Speed-up Scientific Codes

- **Directives (OpenACC)**
- **Libraries**
- **CUDA (or OpenCL)**

3 Way to Accelerate on GPU



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 other codes of the QE distribution

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

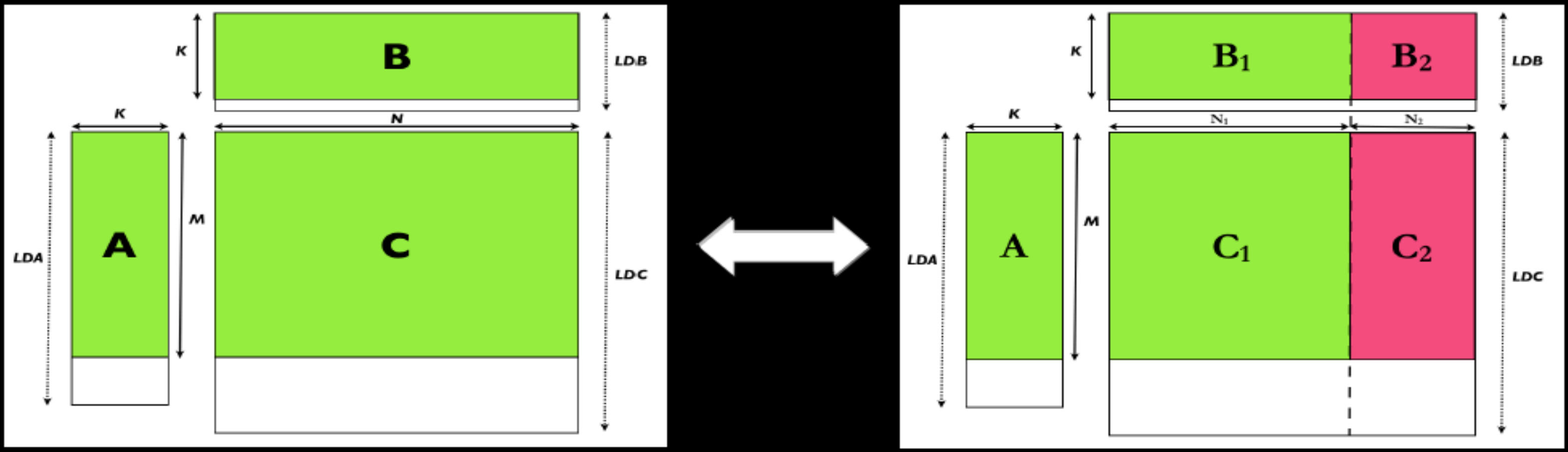
GPU!

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 3DFFTs 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

GPU TECHNOLOGY CONFERENCE

DGEMM: C = alpha A B + beta C



DGEMM(A,B,C) = DGEMM(A,B1,C1) U DGEMM(A,B2,C2)

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

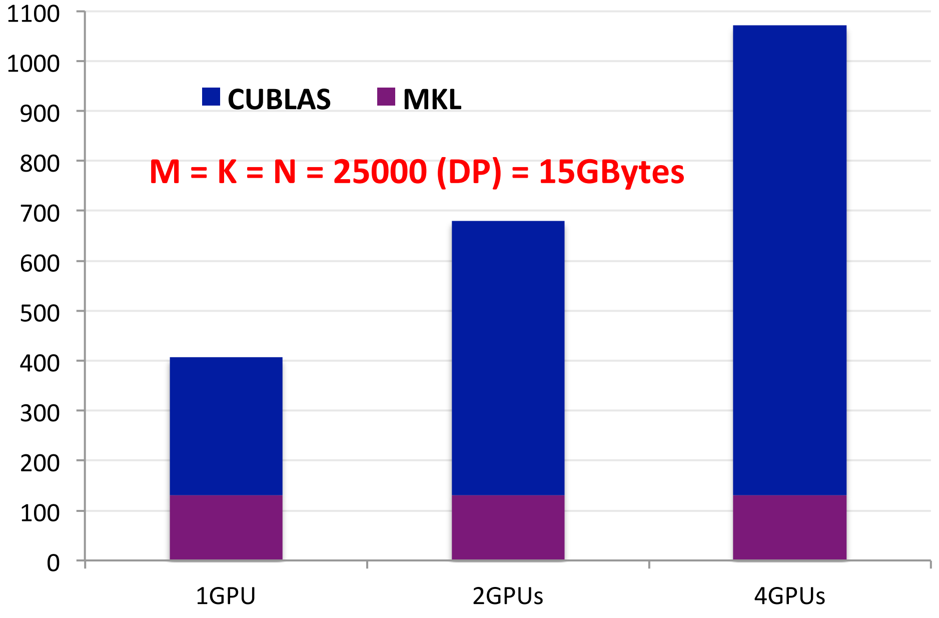
The starting point: ΦGEMM

- [*]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

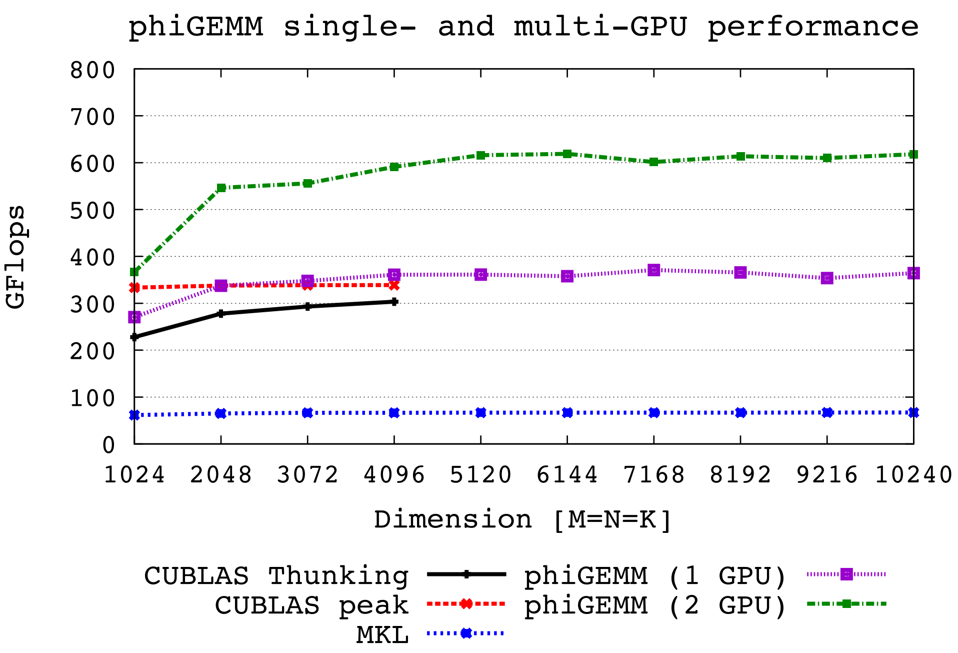
ΦGEMM

http://qe-forge.org/gf/project/phigemm/

ΦGEMM



(2 GPUs x 1 PCI Bus!!)



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 study: no good efficient solutions found so far

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:), &
             nlsm(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
```

MAGMA

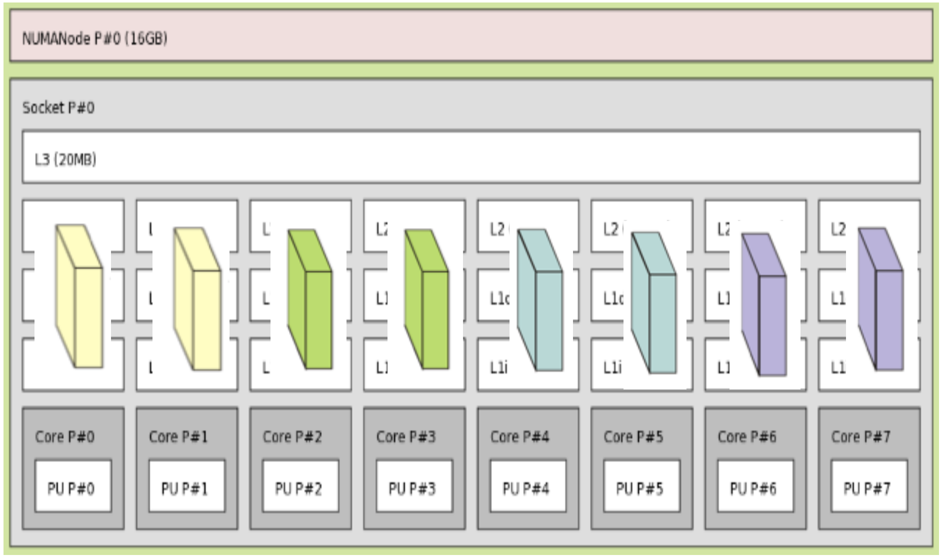
- LAPACK library for GPU systems

```
#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
```

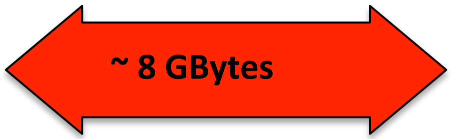
http://icl.cs.utk.edu/magma/

How to use it ...

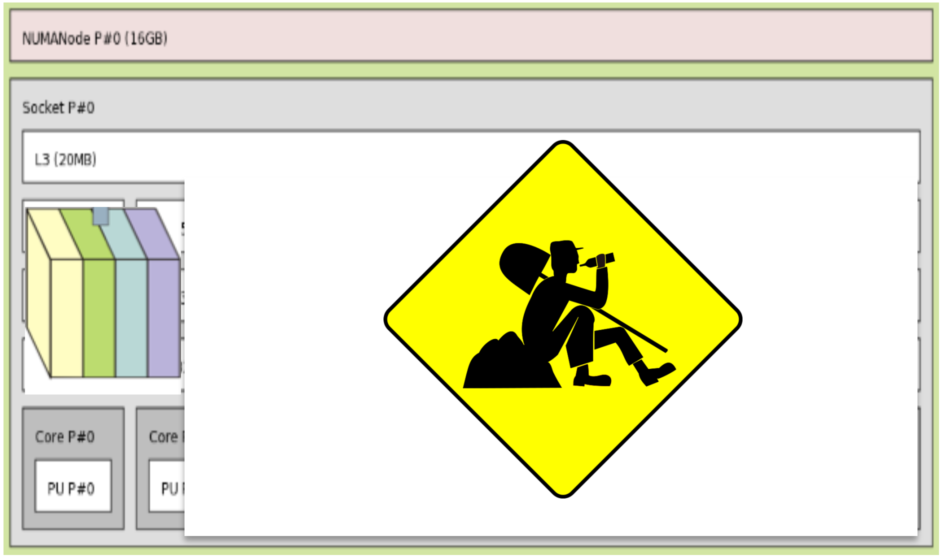
- The partial porting requires to exploit the overall platform: best performances on CPU and GPU
- All complicated balancing issues presented this morning becomes much relevant for the QE-GPU version
- For instance...



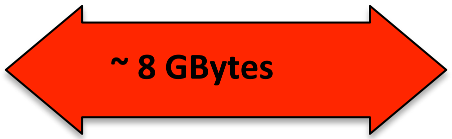
The Intel Xeon E5-2665
Sandy Bridge-EP 2.4GHz



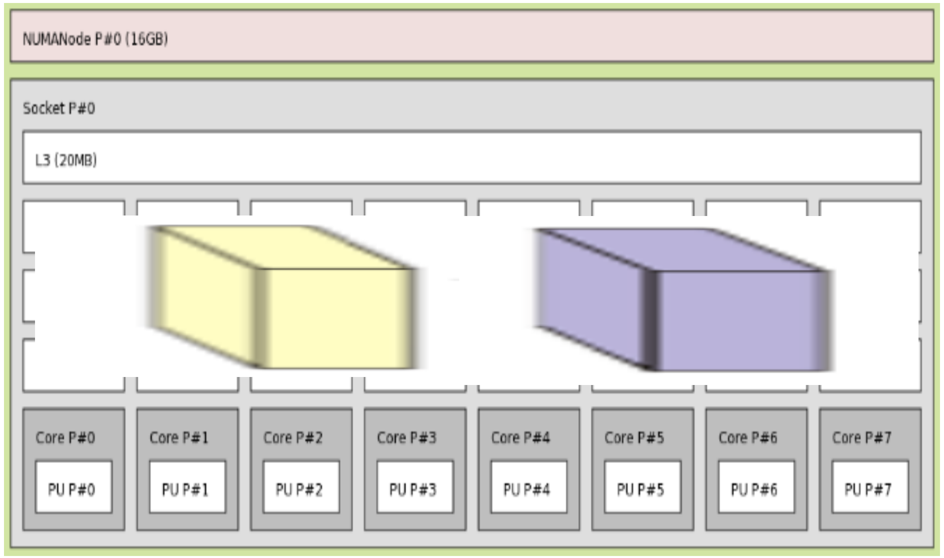
```
mpirun -np 8 pw-gpu.x -inp input file
```

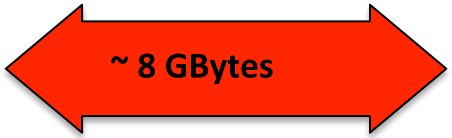


The Intel Xeon E5-2665
Sandy Bridge-EP 2.4GHz



```
mpirun -np 1 pw-gpu.x -inp input file
```

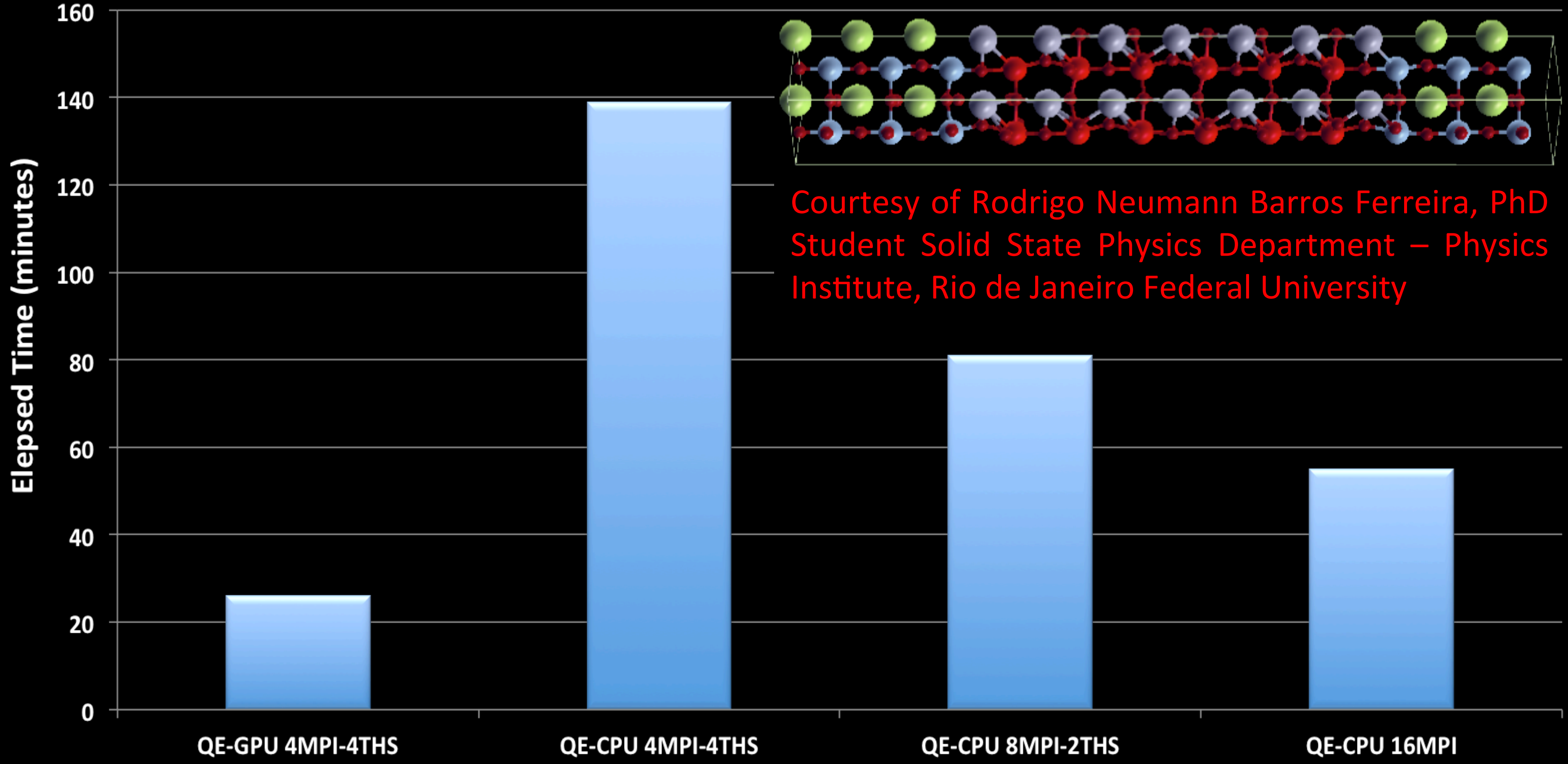


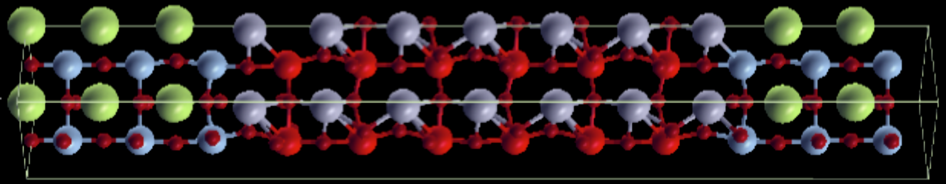


The Intel Xeon E5-2665
Sandy Bridge-EP 2.4GHz

```
export OMP_NUM_THREADS=4
export OPENBLAS_NUM_THREADS=$OMP_NUM_THREADS
mpirun -np 2 pw-gpu.x -inp input file
```

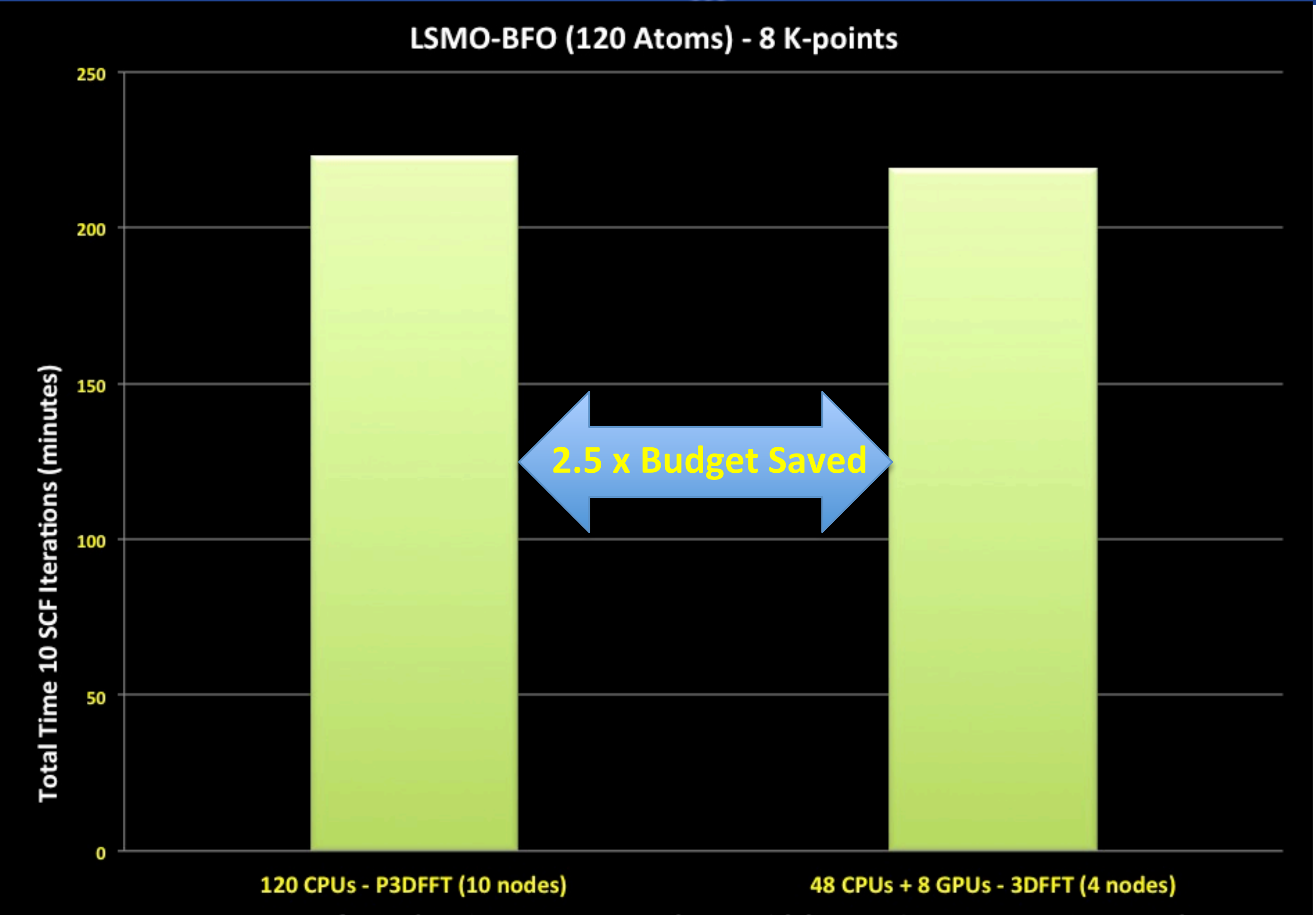
scf calculation (few iterations) on Hydra: LSMO BiFeO3 (BFO)





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

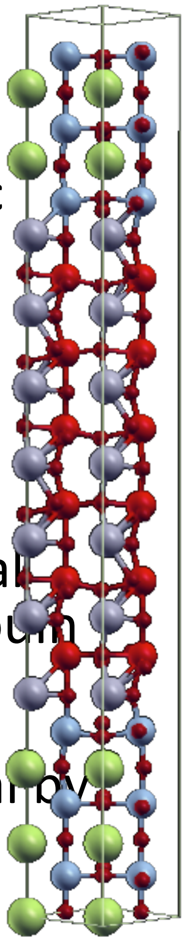
Performance Results /1



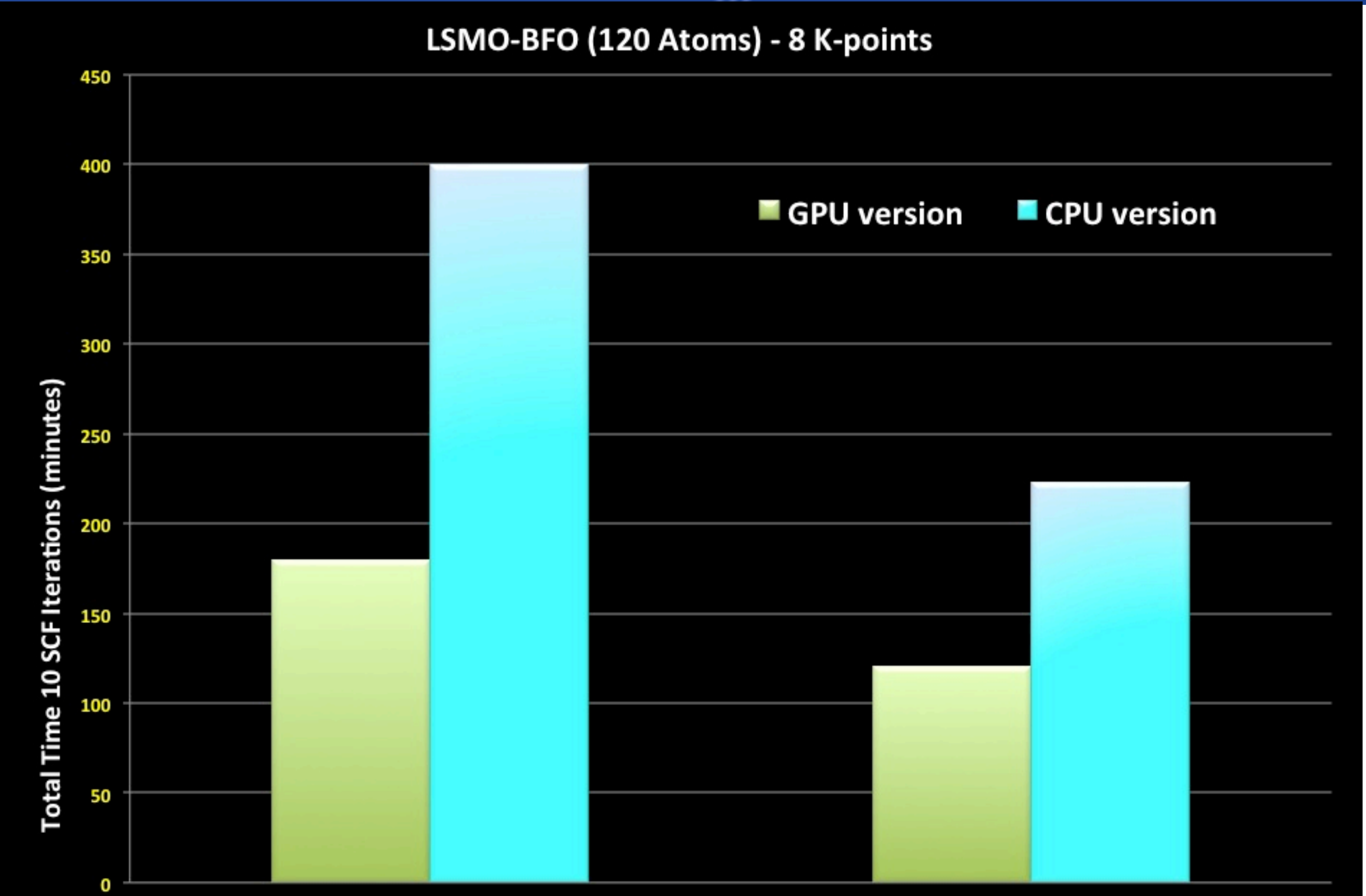
Test performed on CINECA PLX GPU Cluster with best tuning MPI + OpenMP x node
(274 x iDataPlex DX360M2, dual 6-cores Intel Xeon X5550 2.66 GHz + 2 NVIDIA M2070)

Best Practice /1

- Scientific case: La_{2/3}Sr_{1/3}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`



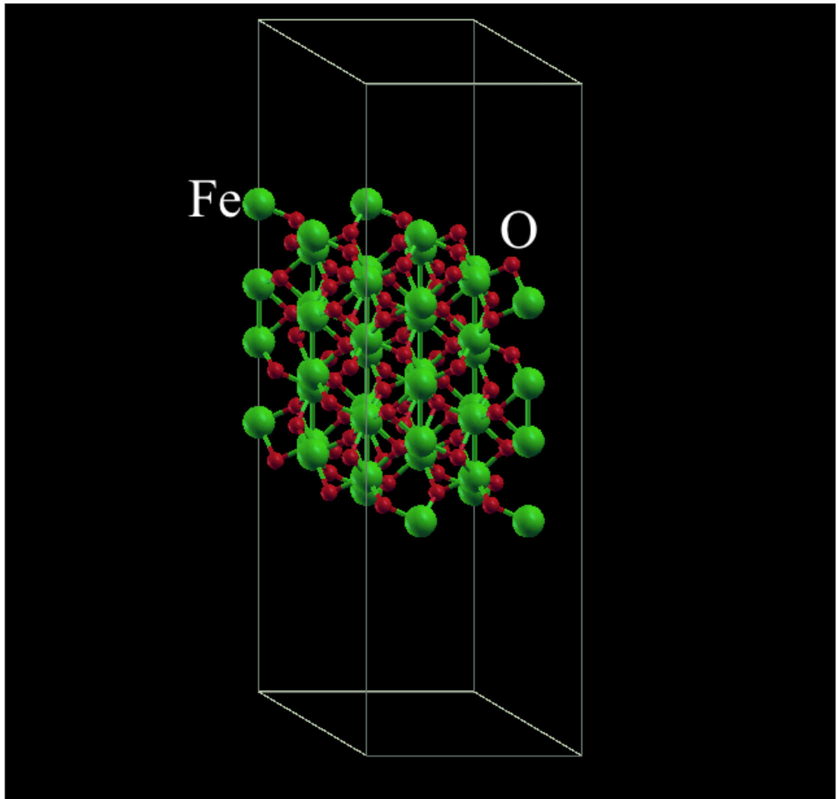
Performance Results /2



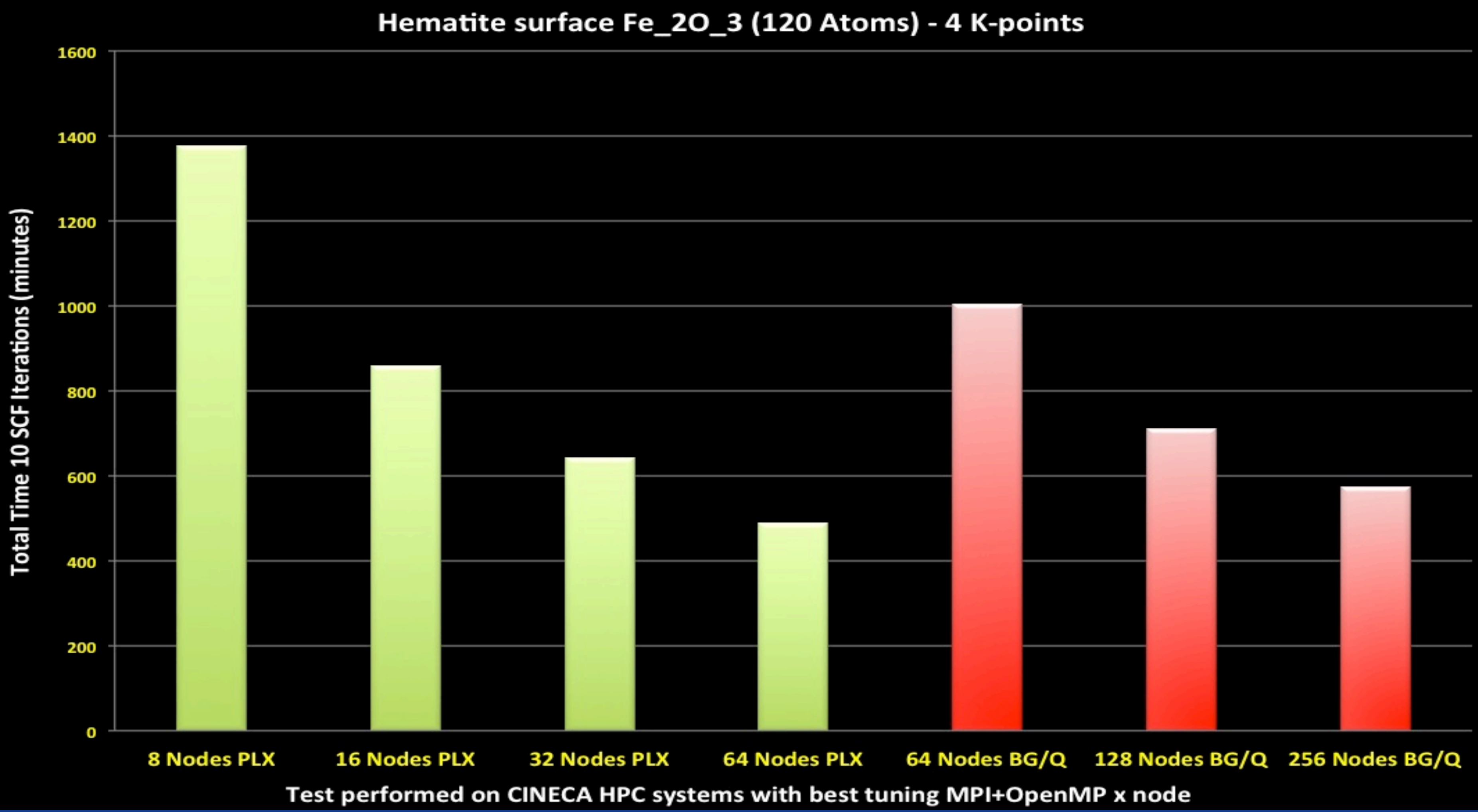
Test performed on CINECA PLX GPU Cluster with 4 MPI and 3Ths x node
(274 x iDataPlex DX360M2, dual 6-cores Intel Xeon X5550 2.66 GHz + 2 NVIDIA M2070)

Best Practice /2

- Scientific case:
  Fe_2O_3 (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(0001): (2x2) surface unit cell


Hematite surface Fe_2O_3 (120 Atoms) - 4 K-points
Total Time 10 SCF Iterations (minutes)
0
200
400
600
800
1000
1200
1400
1600
8 Nodes PLX
16 Nodes PLX
32 Nodes PLX
64 Nodes PLX
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

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
- Feasibility study for improvement of QE-GPU version for other codes the (CP, PH) and EXX section of the PWscf code
- Porting on other accelerated platforms (i.e., Intel-MIC?)

Acknowledgements:

- **Filippo Spiga (Cambridge University/QE-Foundation)**
- 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)
- Mike Atambo (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