Quantum Espresso developers' meeting

Fabio AFFINITO - CINECA
(f.affinito@cineca.it)
Part I:
- Cineca as Intel Parallel Computing center: the effort on Quantum Espresso

Part II:
- Development of the new distribution of g-vectors
Cineca as a "Intel Parallel computing center"

Intel Parallel Computing Centers are universities, institutions, and labs that are leaders in their field. The centers are focusing on modernizing applications to increase parallelism and scalability through optimizations that leverage cores, caches, threads, and vector capabilities of microprocessors and coprocessors. Code modernization is expected to enable large performance increases while maintaining the code portability users expect. By enabling the advancement of parallelism, the Intel® Parallel Computing Centers will accelerate discovery in the fields of energy, finance, manufacturing, life sciences, weather, and beyond.

Cineca is now a Intel® Parallel Computing Center and, in the initial project, the parallelization of codes like Quantum Espresso as well as SPECFEM3D are the target.
Xeon Phi in a nutshell

~60 cores per card
x86 architecture
512-bit SIMD

Differently from GPUs it relies on x86 architecture. It allows to use standard programming models such as MPI and OpenMP within an extended offload language (based on pragma directives). Efficiency is reached only when parallelism and vectorisation are well exploited.
Target: porting of QE on Xeon Phi

- Idea: use the same "strategy" of phiGEMM to run massively parallel portion of code (linear algebra or other) on the MIC architecture (zphizgemm)

- Keep focus on the "maintanability" of the code using as much as possible plug-ins and libraries

- Obtain a code that is suitable for Intel Xeon Phi architecture, but in general for upcoming many cores architectures (in principle different from GPUs)

*QE-MIC project on the QE-Forge has just started*.
ZGEMM('N', 'N', ....)

Blocks are extracted from the matrices A, B and C with Fortran reshapes into the host buffers

Need to take care of the partial blocks at the end
ZGEMM('C','N', ....)

Xphizgemm also needs to deal with hermitian conjugate and transposed matrices. In this case, the blocking is done in the transposed way.
**xphizgmemm**

Fully configurable per MPI process:
- QE_MIC_BLOCKSIZE_M
- QE_MIC_BLOCKSIZE_N
- QE_MIC_BLOCKSIZE_K
- QE_MIC_OFFLOAD_DEVICE
- QE_MIC_OFFLOAD_THRESHOLD

Offload device:
Ensures optimal operation across systems with more than one KNC

Threshold:
ZGEMM to deliver host performance for small and KNC performance for larger matrices
Performance indicators

- Depending on the workload, xphizgemm delivers up to 650GFLOPS/card in QE
- For small and odd-shaped matrices a considerable performance drop can be observed
- Final write back of c-blocks is still not async, hence there is an opportunity for improvement still
The old tricky way...
The new distribution
Some improvement...
Something that we expect to improve soon...
Something unexpected, but we don't complain :)

PLX 24 core
Water 32
> New Data Distro
> Old Data Distro
CONS:

- less balanced
- some processors could have zero vectors (now that produces a crash..)

PROS:

- easier to manage
- it permits to implements third-parties components
- increase data locality
- more efficient when FFT uses all-to-all
Acknowledgements:

- Chris Dahnken (INTEL)
- Eric Pascolo (Unimore)