Domain specific library for electronic structure calculations

M. Taillefumier, A. Kozhevnikov, I. Sivkov, J. VandeVondele, T. C. Schulthess

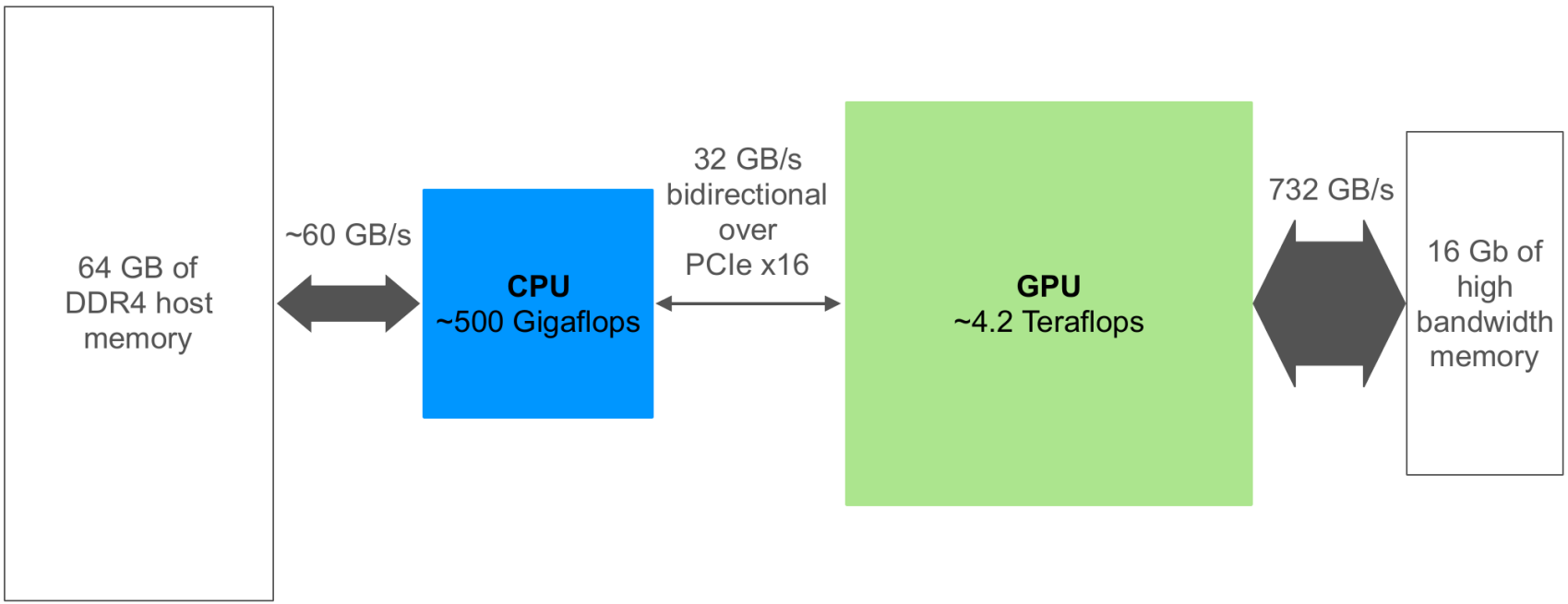
ETH Zurich / CSCS (Lugano), Switzerland

February 16, 2018

Piz Daint at CSCS

Current configuration

Cray XC50, 5320 nodes : Intel Xeon E5-2690v3 12C, 2.6GHz, 64GB + NVIDIA Tesla P100 16GB 4.761 Teraflops / node



Best perfs

Optimize code that fully use GPUs.

Programs for electronic structure code

Most of these programs are older than i do or close to

Basis functions for KS states / Atomic potential treatment	Periodic Bloch functions (plane-waves or similar)	Localized orbitals
Full-potential	FLEUR Wien2K Exciting Elk	FHI-aims FPLO
Pseudo-potential	VASP CPMD Quantum ESPRESSO Abinit Qbox	CP2K SIESTA OpenMX

Problems

too many of them do not perform well on hybrid architecture if not at all

Porting code to GPUs : No silvers bullets

Usually involve several steps

- refactor the code and identify bottlenecks
- change data layout
- optimize CPUs threads and prepare code for node level parallelization
- then move compute intensive kernels to GPUs.

Problems

- We have to do that for all major codes. Not enough men power for that
- working on these codes is difficult.

Porting code to GPUs : Additional problems

on the GPU side

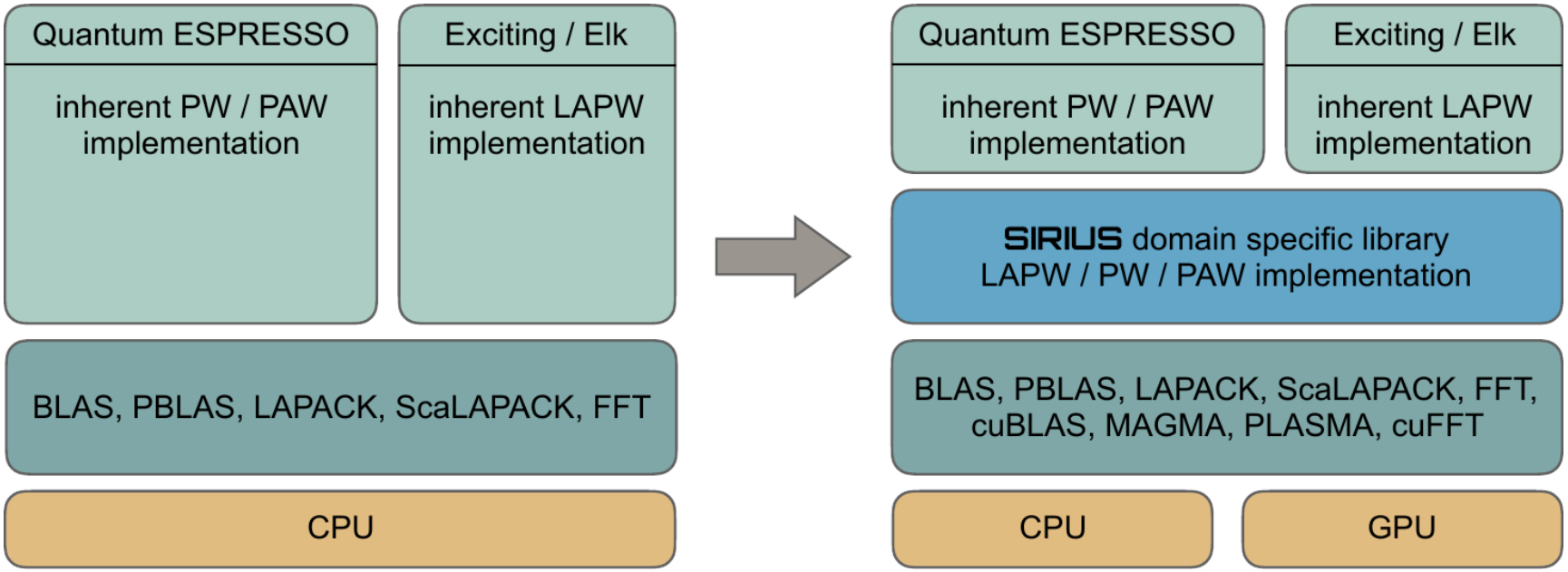
- Too many ways to program GPUs : CUDA, OpenCL, OpenACC which one to choose ?
- combine GPUs with MPI and openmp
- Many supercomputers have multi-gpu per node (Summit)

On the developer side : Two options

1. Scientists are not computer scientists so it can represent a huge challenge for them.
2. Software ingeneers will also have difficulties with the base codes (lack of clear structure most of the time).

How can we solve this cannundrum ?

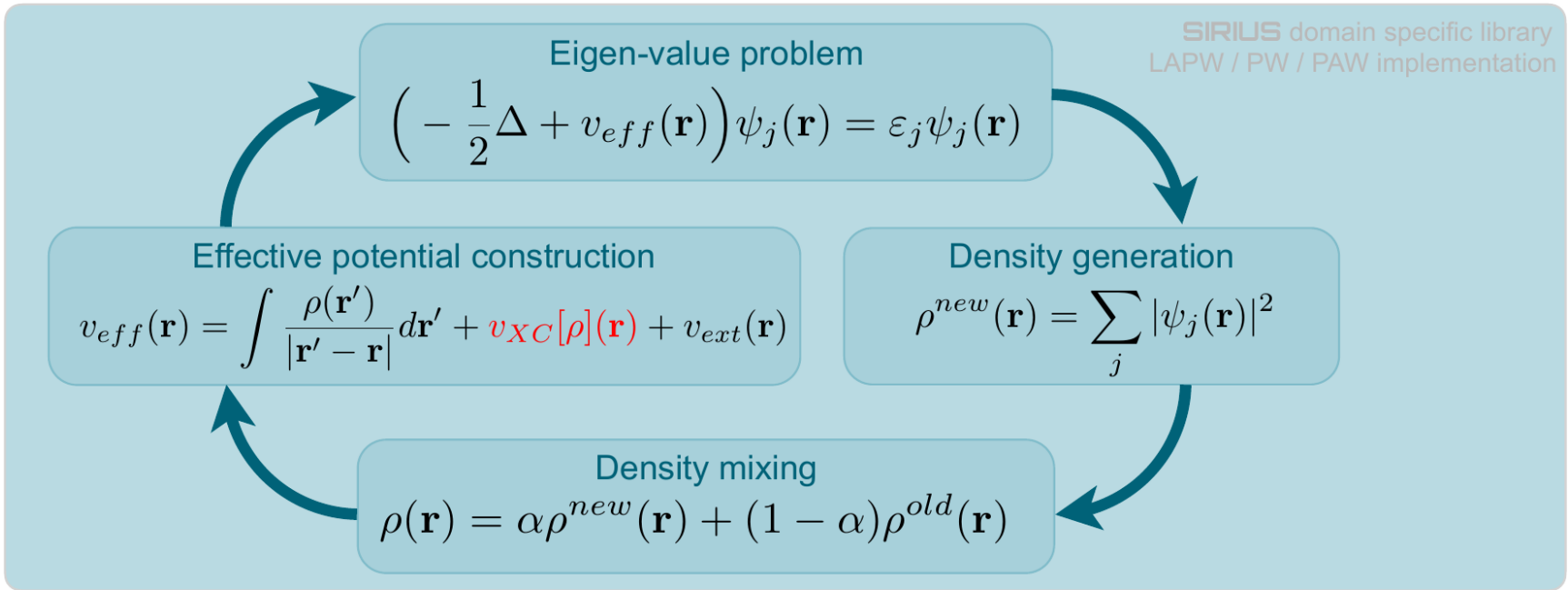
Extend the legacy Fortran codes with the API calls to a domain-specific library which runs on GPUs and other novel architectures.



Advantages

- Scientists can contribute to new features while software ingeneers can work on the different backends
- easier maintenance and portability to new architectures

Where to draw the line : Calculation of the ground state



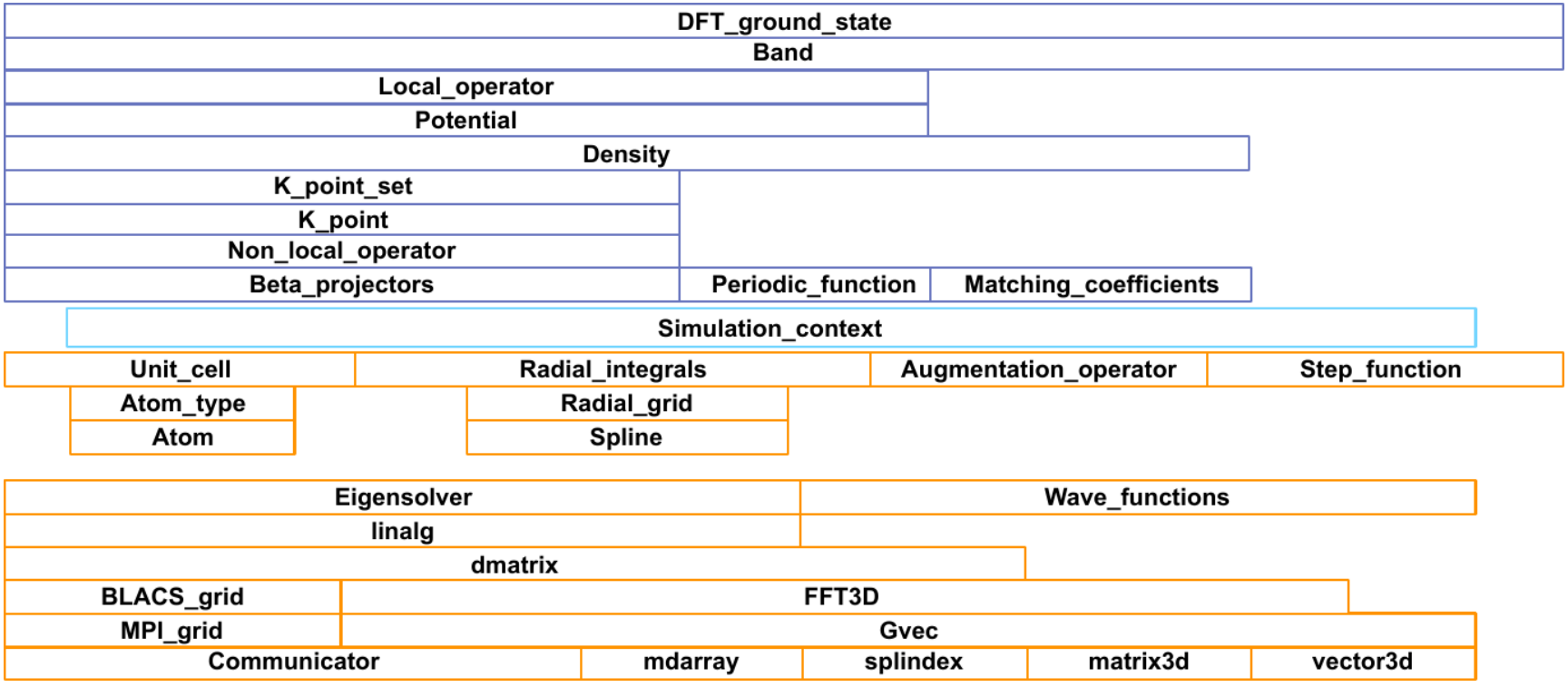
Outputs

- Wavefunctions and energies
- charge density and magnetization
- force and stress tensors
- We have everything for the rest of the calculations

SIRIUS Library

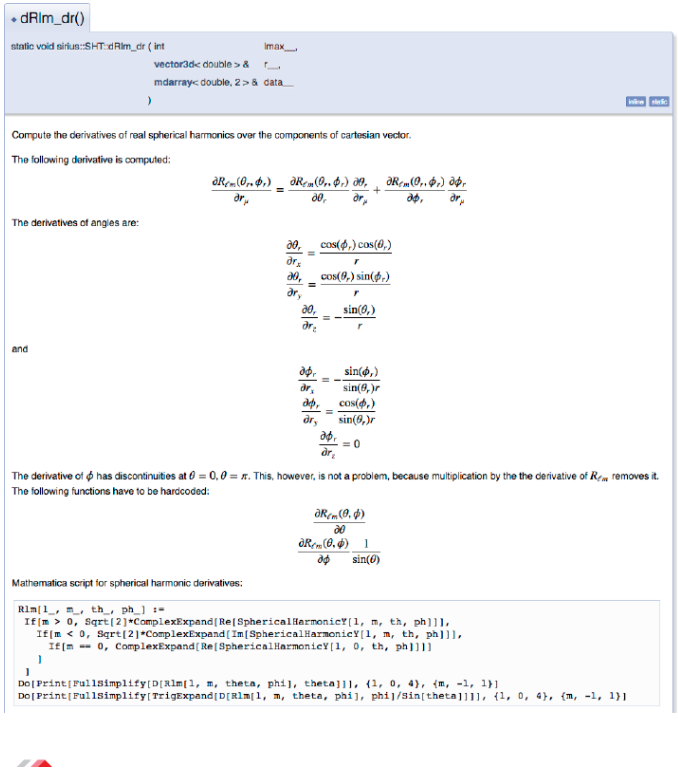
https://github.com/electronic-structure/SIRIUS

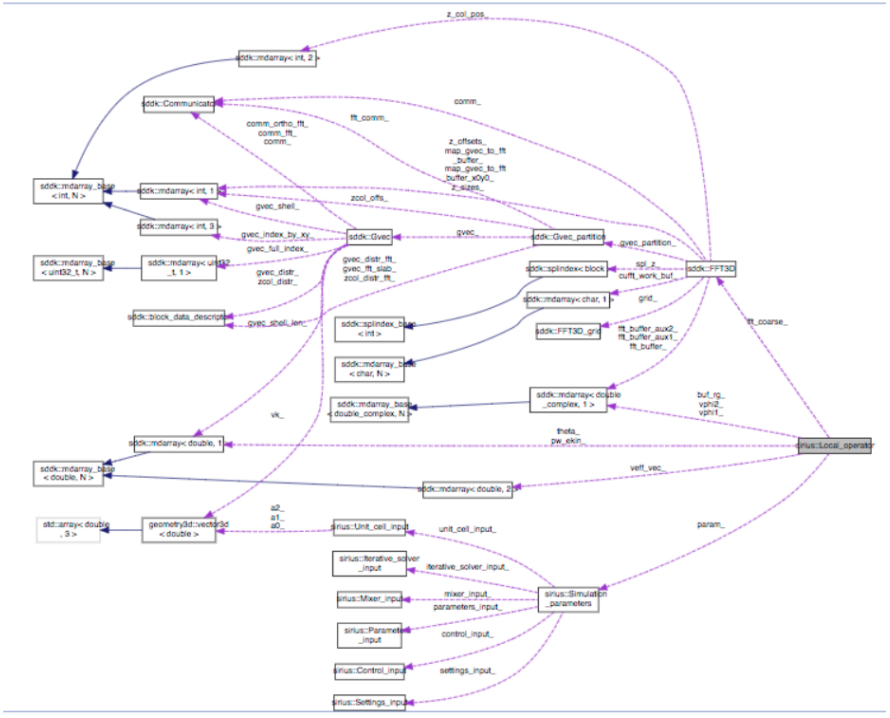
SIRIUS is a collection of classes that abstract away the different building blocks of PW and LAPW codes. The class composition hierarchy starts from the most primitive classes and progresses towards several high-level classes. The code is written in C++11 with MPI, OpenMP and CUDA programming models.



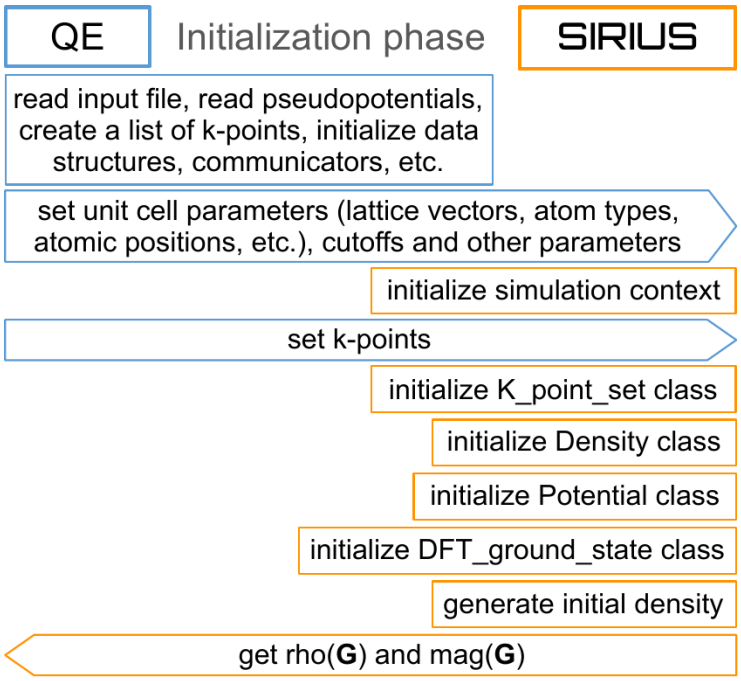
Doxygen documentation

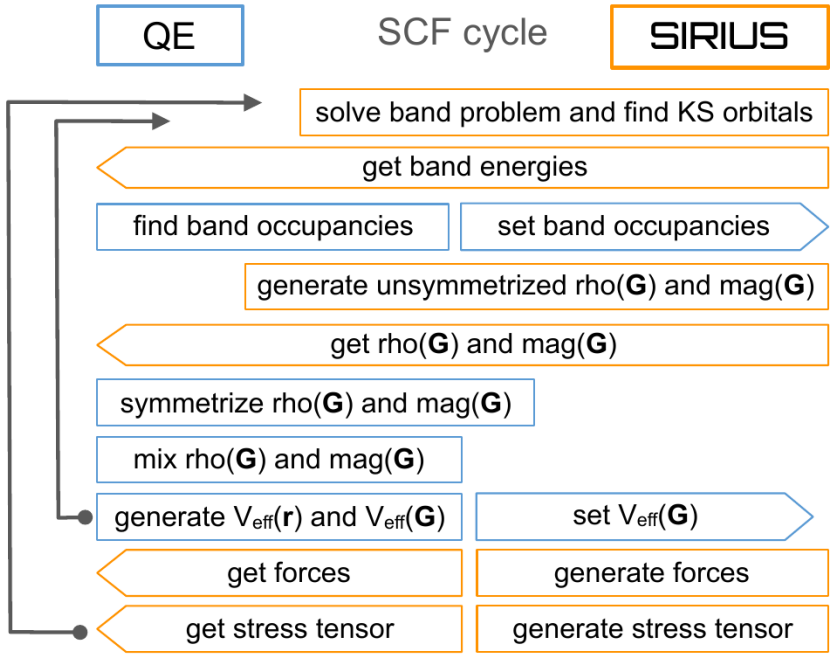
https://electronic-structure.github.io/SIRIUS-doc/





Example of interoperability : Quantum expresso + SIRIUS



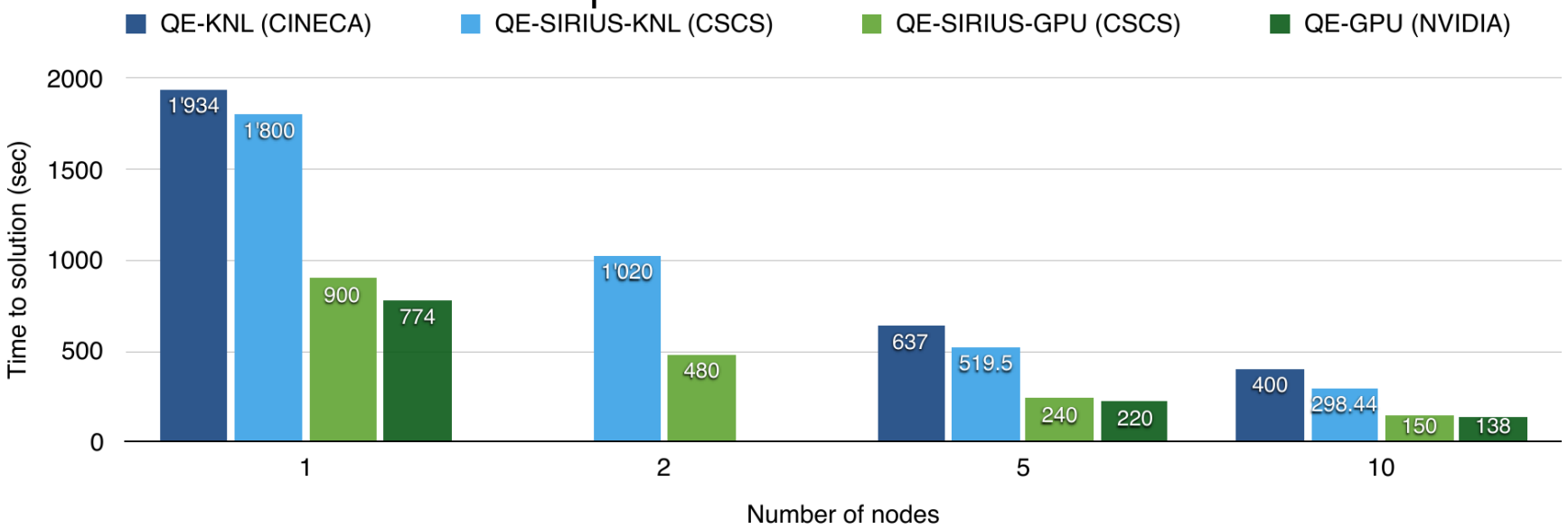


Goals

Really do the most intensive computing parts on GPUs

Variable cell relaxation of Si63Ge

Performance benchmark of the QE, Cuda Fortran version of QE and SIRIUS-enabled QE codes for the 64-atom unit cell of $Si_{1-x}Ge_x$. The runs were performed on hybrid nodes with 12-core Intel Haswell @2.5GHz +NVIDIA Tesla P100 card (QE-GPU, QE-SIRIUS-GPU) and on nodes with 68-core Intel Xeon Phi processor @1.4 GHz (QE-KNL). Time for the full vc-relax calculation is reported.



Ground state of Pt-cluster in water

Performance benchmark of the QE and SIRIUS-enabled QE codes for the 288-atom unit cell of Pt cluster embedded in water. The runs were performed on dual socket 18-core Intel Broadwell @2.1GHz nodes (BW), on hybrid nodes with 12-core Intel Haswell @2.5GHz + NVIDIA Tesla P100 card (GPU) and on nodes with 64-core Intel Xeon Phi processor @1.3 GHz (KNL). ELPA eigen-value solver was used for CPU runs. Time for the SCF ground state calculation is reported.