DESIGNING A HIGH PERFORMANCE AND PORTABLE LIBRARY (QMCKL): ONE OF THE MAJOR CHALLENGES ADDRESSED BY TREX CoE

Quantum Monte Carlo (QMC) methods have many application areas, ranging from boosting catalyst design for sustainable fuel production to the simulation of new materials with high critical superconducting temperature. For some large-system applications, QMC methods represent the only possibility to compute high-accuracy properties. However, these methods are extremely CPU intensive and very often require Exascale computing capabilities. Fortunately, their embarrassingly parallel nature facilitates the use of a large number of nodes. However, efficiently using the computational power of recent multicores and accelerators remains a major challenge.

To address this issue, the TREX (Targeting Real Chemical Accuracy at the EXascale) CoE (https://trex-coe.eu/) has chosen to develop a high performance portable library QMCKL (see https://trex-coe.github.io/qmckl/index.html for documentation)

This library aims at fulfilling three (classical) goals:

1) Productivity: making sure that the library can be easily used by chemists and integrated into their code (in particular the TREX reference codes)
2) Performance: offering high level performance level
3) Portable: being available on a large number of platforms

Simultaneously reaching these three goals is rather challenging. We will discuss here how, in QMCKL, we are tackling these issues, namely,

- by making available multiple library versions: a reference one easy to integrate and understand and optimized versions for the main reference targets;
- by making tradeoffs between performance and portability;
- by using high level algorithm descriptions to encompass multiple optimizations (vectorization, register allocation) common to different architectures;
- by relying on different tools such as auto-tuners to get the “last mile” in performance; by taking into account numerical accuracy issues allowing the developer to use lower precision (including FP16 format) code version when possible as a tradeoff for performance. Due to the Monte Carlo nature of the algorithm, QMC methods can accommodate (for some part) lower precision computation which can provide substantial performance gains.

We will demonstrate QMCKL use within QMC=CHEM an important QMC code and we will show performance analysis on different target architectures.

These developments are first supported by the use of MAQAO, an advanced performance analysis framework (www.maqao.org) which identifies profitable optimizations (partial/full vectorization, simplifying control flow, low iteration count, data access restructuring, blocking/interchanging, load balancing etc….) and performs a Return on Investment (ROI) analysis to help the developer select the most profitable optimization. Furthermore, MAQAO performs automatically comparative runs to analyze compiler impact, various datasets, different algorithms and different configurations (number of cores, etc..). Numerical accuracy work is supported by the use of VERIFICARLO (https://github.com/verificarlo/verificarlo), a tool for debugging and assessing floating point precision and reproducibility.