LAWRENCE
LIVERMORE
NATIONAL
LABORATORY

LLNL-TR-705122

CDAC Student Report: Summary of LLNL Internship

J. E. Herriman

October 12, 2016

Disclaimer

This work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.

CDAC Student Report: Summary of LLNL internship

Jane E. Herriman
October 10, 2016

IM release number: LLNL-TR-705122

1 Introduction

Multiple objectives motivated me to apply for an internship at LLNL: I wanted to experience the work environment at a national lab, to learn about research and job opportunities at LLNL in particular, and to gain greater experience with code development, particularly within the realm of high performance computing (HPC).

This summer I was selected to participate in LLNL's Computational Chemistry and Material Science Summer Institute (CCMS). CCMS is a 10 week program hosted by the Quantum Simulations group leader, Dr. Eric Schwegler. CCMS connects graduate students to mentors at LLNL involved in similar research and provides weekly seminars on a broad array of topics from within chemistry and materials science. Dr. Xavier Andrade and Dr. Erik Draeger served as my co-mentors over the summer, and Dr. Andrade continues to mentor me now that CCMS has concluded. Dr. Andrade is a member of the Quantum Simulations group within the Physical and Life Sciences at LLNL, and Dr. Draeger leads the HPC group within the Center for Applied Scientific Computing (CASC). The two have worked together to develop Qb@ll, an open-source first principles molecular dynamics code that was the platform for my summer research project.

2 Research experience at LLNL

2.1 Intro: Qb@ll and Ehrenfest molecular dynamics

Qb@ll is a first principles molecular dynamics (first principles MD) code freely available at `https://github.com/LLNL/qball`. Qb@ll is designed for scalability, allowing it to take advantage of today's increasingly parallel architectures.

Qb@ll offers both Born-Oppenheimer MD and Ehrenfest MD. In both types of first principles MD, atomic/ionic positions are calculated from their positions

and forces at the previous time step; these forces are obtained from the electron wave function. In Born-Oppenheimer MD, the electron wave function is calculated via density functional theory (DFT) and in Ehrenfest MD, the electron wave function is calculated from time-dependent density functional theory (TDDFT). Therefore, Born-Oppenheimer MD relies on the adiabatic principle, which assumes that from the point of view of the ions, the electrons are always in the ground state. This assumption is reasonable when the time scales of the ions and electrons are very different. If the electrons are moving very fast relative to the ions, we can decouple their motion. In Ehrenfest MD, however, this assumption is lifted. The electron wave function can evolve out of the ground state, allowing us to model excited state phenomena like the exposure of a material to an electric field or the impact of high-velocity particle radiation on a target.

To calculate the evolution of the electron wave function, we need a propagator — something that integrates the time dependent Kohn-Sham equations (an analog to the time dependent Schrodinger equation) in time, allowing us to obtain the electron wave function at one time step, given the wave function at the preceding step. The goal of my research at LLNL has been to develop better methods of time integration in Qb@ll. My efforts have thus far been focused on the Enforced Time Reversal Symmetry (ETRS) propagator, which will be explained in the next section.

2.2 ETRS propagator

The Enforced Time Reversal Symmetry (ETRS) propagator exhibits greater energy and charge conservation than comparably expensive Fourth-order Runge Kutta methods; as the name suggests, the strengths of this propagator, including this enhanced conservation, draws from time reversal symmetry. ETRS introduces an effective coupling from one time step to the next that creates greater consistency over the course of a simulation.

The ETRS propagator is built using the principle that, if propagating by some time step Δt takes us from $\phi(t)$ to $\phi(t + \Delta t)$, we should similarly be able to propagate 'backwards' in time by a time step of $-\Delta t$ and recover $\phi(t)$ from $\phi(t + \Delta t)$.

Mathematically this may be expressed as

$$\exp(i\Delta t\mathbf{H}(t + \Delta t))\phi(t + \Delta t) = \phi(t) \tag{1}$$

given that

$$\exp(-i\Delta t\mathbf{H}(t))\phi(t) = \phi(t + \Delta t) \tag{2}$$

.

From these statements we derive the condition

$$\exp(i\frac{\Delta t}{2}\mathbf{H}(t + \Delta t))\phi(t + \Delta t) = \exp(-i\frac{\Delta t}{2}\mathbf{H}(t))\phi(t) \tag{3}$$

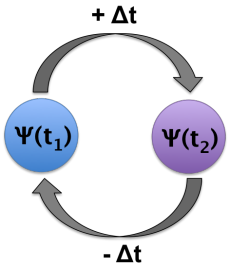

Figure 1: Visual illustration of ETRS

Rearranging, we see get

$$\phi(t+\Delta t) = \exp(-i\frac{\Delta t}{2}\mathbf{H}(t+\Delta t))\exp(-i\frac{\Delta t}{2}\mathbf{H}(t))\phi(t) \tag{4}$$

which allows us to define our ETRS propagator

$$\mathbf{U}_{ETRS}(t+\Delta t,t) = \exp(-i\frac{\Delta t}{2}\mathbf{H}(t+\Delta t))\exp(-i\frac{\Delta t}{2}\mathbf{H}(t)) \tag{5}$$

such that

$$\phi(t+\Delta t) = \mathbf{U}_{ETRS}(t+\Delta t,t)\phi(t) \tag{6}$$

.

This is all well and good, but we immediately see a difficulty: In order to get the wavefunctions at time $t+\Delta t$ we need the electron density at time $t+\Delta t$ in order to construct the Hamiltonian $\mathbf{H}(t+\Delta t)$. In practice, we begin by approximating $\mathbf{H}(t+\Delta t)$ to get the wavefunction; we then use this approximate wavefunction to construct an improved approximation to $\mathbf{H}(t+\Delta t)$. In the version of the ETRS propagator implemented in Qb@ll, we simply use $\mathbf{H}(t)$ as an approximation to $\mathbf{H}(t+\Delta t)$ on the first iteration and then use the wavefunction $\phi_{approx}(t+\Delta t)$ calculated via

$$\phi_{approx}(t+\Delta t) = \exp(-i\Delta t\mathbf{H}(t))\phi(t) \tag{7}$$

to construct $\mathbf{H}(t+\Delta t)$.

2.3 Refactoring the implementation of ETRS in Qb@ll

In practice, we approximate the action of the matrix exponentials of which the ETRS propagator $\mathbf{U}_{ETRS}(t+\Delta t,t)$ is composed by their fourth order Taylor series. For a matrix $\mathbf{A}$ and wavefunction ϕ, $\exp(A)\phi$ is approximated as

$$\exp(\mathbf{A})\phi = \phi + \mathbf{A}\phi + \frac{\mathbf{A}^2}{2}\phi + \frac{\mathbf{A}^3}{6}\phi + \frac{\mathbf{A}^4}{24}\phi \tag{8}$$

via a call to a function in qb@ll called '*exponential*()'.

We recognized that refactoring the code and changing the interfaces of some of the objects therein would allow us to express three distinct calls to *exponential*() as two calls. For each time step, propagating the wavefunction via $\mathbf{U}_{ETRS}(t+\Delta t, t)\phi(t)$ now requires two rather than three *exponential*() calls. Therefore, if all of the time required to simulate a time step were spent in calls to *exponential*(), we would expect to see a 50% speedup from this refactorization

2.4 Performance

2.4.1 Performance metrics

The performance metric used in this work is simulated/wall time (fs/hour), i.e. the number of femtoseconds that the user can simulate in Qb@ll in one hour of the user's time. Qb@ll's performance for a given number of nodes was measured using the optimal mapping of tasks to processors, explained below.

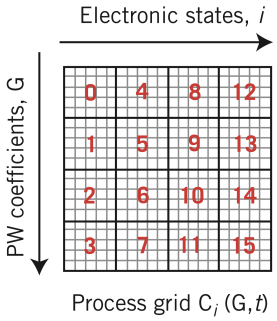

Figure 2: A schematic of Qb@ll's 2D MPI logic grid, taken from [1].

When a system is simulated in Qb@ll, each of the electronic states (orbitals) in the system is expressed as a linear combination of plane waves. The system wavefunction is discretized on a 2D logic grid, as shown in Figure 2, which stores the coefficients by plane wave and by electronic state. The 2D logic grid is then partitioned amongst MPI processes according to a user directive. (Typically one MPI process is forked on each node.) The number of ways to partition the 2D logic grid into N subdomains increases with N, and not all partitions result in equally good performance. We therefore tested all possible grid partitions for a given number of processors and measured the performance of the ETRS propagator for a given number of nodes using the optimal partition for that number of nodes.

2.4.2 Performance increases from refactoring the ETRS implementation

As mentioned in a previous section, performance increases from refactoring the ETRS propagator as described above must be no greater than 50%. We demon-

strated the impact of the refactorization on up to 32768 processors; in practice, we saw speedups of approximately 35% to 20% for the range of processors tested, where the speedup decreased as the number of processors used increased. This performance increase is captured in Figures 3 and 4.

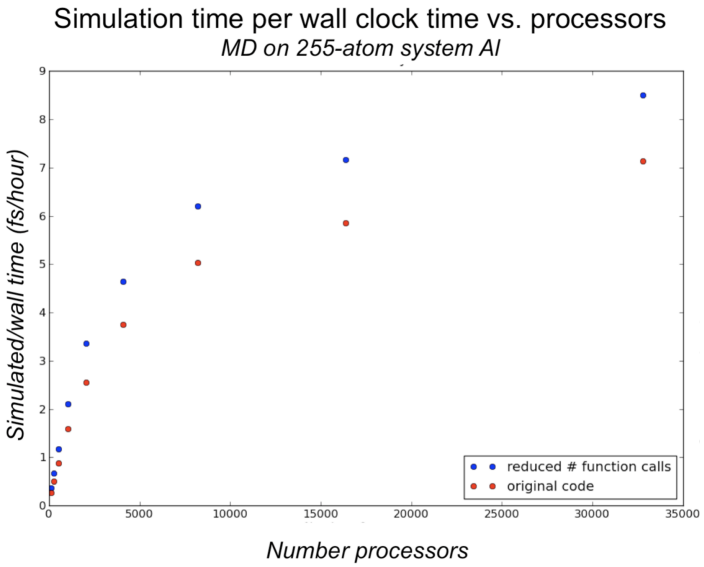

Figure 3: Above is plotted a performance metric – simulated time over wall time in fs/hour – versus the number of processors. The red curve shows the scaling of the ETRS propagator in Qb@ll in its initial implementation; the blue curve shows performance scaling for the refactored code, where two calls to *exponential*() have been condensed. The upward shifting of the blue curve relative to the red shows the associated performance increase.

2.5 Ongoing work

We are interested in increasing the performance of the ETRS propagator by (1) decreasing the computational cost of performing a time step and/or (2) increasing the stability of the propagator to allow for longer time steps. To these ends, we are considering multiple paths forward.

Of particular interest in changing the way that we calculate the action of the matrix exponential. A vast body of literature exists on this topic. Krylov subspace projections, Pade approximants, and Scaling and Squaring methods are among those techniques in the literature that, at least in some contexts, are reported to outperform Taylor series approximations.

We are also interested in trying implicit rather than explicit methods, which, though more computationally intensive, can result in more stable propagation.

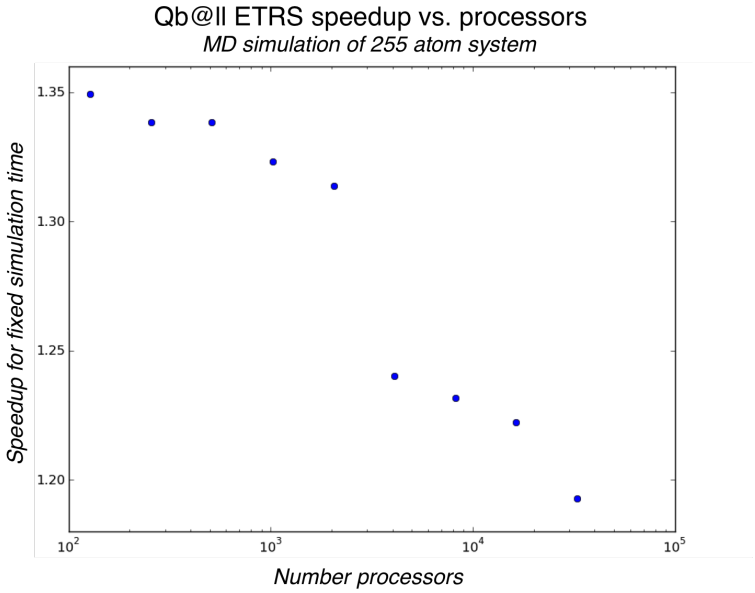

Figure 4: Qb@ll's speedup from refactoring the ETRS propagator to rely on less calls to *exponential*() is plotted versus the number of processors. The speedup is quantified as the simulated/wall time in the refactored code divided by the simulated/wall time in the original code; it therefore may be obtained by dividing the blue points in Figure 3 by their associated red points.

3 Broader professional development at LLNL

While at LLNL, networking with LLNL and Sandia staff scientists has been hugely rewarding. Meeting scientists engaged in different aspects of computational research — from computational physics to scientific computing to software-hardware co-development — has provided me with a wealth of information and helped me to refine my career goals. I now have a better sense of the sorts of opportunities that exist within the national lab system, and at LLNL in particular. I have learned about the wide array of research problems addressed by LLNL scientists, the structure of LLNL, i.e. where to go looking for positions that could benefit from my skill set, and about different revenue streams that can be used to pursue important research questions. Exposure to many career trajectories within the national lab system has helped me to identify positions in which I might like to find myself after graduate school and skills that I should develop to be competitive for these positions.

4 Conclusion

My experience at LLNL thus far has been very worthwhile. Already I have expanded my view of post-graduation career opportunities, gained greater exposure to software development and research questions within HPC, and benefited from the mentorship of individuals who are experts in the areas I would ultimately like to work. I look forward to continuing my thesis research from here and taking advantage of the many opportunities the lab has to offer.

References

[1] A. Schleife et al. Quantum dynamics simulation of electrons in materials on high-performance computers. *Computing in Science & Engineering*, 2014.

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.