I am a first year CS graduate student researching high performance algorithms, applications, and architectures. I am currently mainly working with Prof James Demmel on parallel numerical linear algebra algorithms. In the past, I have worked at the Parallel Programming Labrotary at University of Illinois, where I was advised by Prof Laxmikant Kale and worked on the development of Charm++, Charm++ applications, as well as general high performance algorithms and applications. I have also contributed to the development of a multi-GPU version of Desmond, a parallel molecular dynamics application, at DE Shaw Research.

1 MPQC: Massive Parallel Quantum Chemistry

MPQC is a parallel software for ab initio quantum chemistry simulation, which computes properties of molecules using the laws of quantum mechanics. The application provides a number of approximations for solving the molecular Schrodinger equation. The software is targeted at highly parallel architectures and is designed to scale to many thousands of processors.

1.1 Scientific background

MPQC provides support for several approximations to the wave function of the quantum many-body system, the exact solution to which is intractable. Supported methods include the popular Density Functional Theory (DFT) and the Hartree-Fock method. The Hartree-Fock method, in particular, is a building block for many high-accuracy many-body methods such as coupled-cluster theory.

1.2 Fock-matrix construction

The most computationally-heavy kernel in most quantum chemistry applications, including MPQC, and methods is the construction of the Fock matrix. The Fock matrix is constructed to represent the many-body system interactions via the computation of two-electron integrals as the pairwise interaction between pairs of electrons and contracting these with the density matrix, which represents the electron distribution in physical space.

1.3 Parallelization

The Fock matrix construction can be parallelized in various ways and is the topic of on-going research. Typically, in MPQC the matrix is replicated on each processor, which allows for less-frequent synchronization among processors and better load balance. However, the replicated-data
approach does not scale to very high degrees of parallelism and more complex methods need to be employed.

1.4 Performance

MPQC runs on various platforms, however, it has been recently used on and optimized on the Bluegene/P architecture. Figure 1.4 demonstrates recent the improvement of recent optimizations for MPQC as well as the overall application scaling. The scaling results show that the diagonalization of the matrix is currently a major scaling bottleneck. This problem is due to the diagonalization being performed on a small matrix that is distributed over many processors. However, many efficient methods exist for parallel diagonalization and the current performance in MPQC is due to an inefficient implementation, which is currently being fixed. However, the scalability of the Fock matrix build also flattens at high degrees of parallelism and new parallelization methods are currently being considered to fix this problem.

Figure 1: Scalability of MPQC after recent optimization on Intrepid (BG/P) at Argonne National Lab