**Name:** ***Akhil Shajan (Self Project)***

# *OpenMP parallelization of QUICK program*

QUICK is an open source, GPU enabled, *ab initio* and density functional theory program developed by Götz lab at University of California San Diego and Merz lab at Michigan State University. It has different features like Hartree-Fock energy calculations, Density functional theory calculations (LDA, GGA and Hybrid-GGA functionals available), Gradient and geometry optimization calculations and Mulliken charge analysis. It also supports QM/MM calculations with Amber21.

Fortran API is used in QUICK as QM energy and force engine and also MPI parallelization for CPU platforms is also implemented. Massively parallel GPU implementation via CUDA for Nvidia GPUs and Multi-GPU support via MPI + CUDA, also across multiple compute nodes is achieved.

Even though MPI+CUDA gives a very good speed, it is only achieved across multiple node and GPUs. But in cases where user with single node with multiple cores and multiple GPUs and the cores would fight for access to the memory at the same time. By sharing memory, OpenMP threads can reduce the memory footprint of the application, that means faster working code.

OpenMP is an Application Program Interface (API), jointly defined by a group of major computer hardware and software vendors. OpenMP provides a portable, scalable model for developers of shared memory parallel applications. The API supports C/C++ and Fortran on a wide variety of architectures. OpenMP is designed for multi-processor/core, shared memory machines. OpenMP programs accomplish parallelism exclusively through the use of threads. A thread of execution is the smallest unit of processing that can be scheduled by an operating system. The idea of a subroutine that can be scheduled to run autonomously might help explain what a thread is. Threads exist within the resources of a single process. Without the process, they cease to exist. Typically, the number of threads match the number of machine processors/cores. However, the actual use of threads is up to the application. Since QUICK is a large piece of code, OpenMP would be applied first to the Hatree-Fock method and then extended to the rest of the code over time.

In the Hartree–Fock method of quantum mechanics, the Fock matrix is a matrix approximating the single-electron energy operator of a given quantum system in each set of basis vectors. The Fock matrix approximates the quantum system's true Hamiltonian operator. It only incorporates the effects of electron-electron repulsion on a mediocre level. It is also worth noting that, being a one-electron operator, the Fock operator excludes the electron correlation energy.

The Fock matrix is defined by the Fock operator. The Fock operator for the *i-th* electron is provided for the limited situation, which assumes closed-shell orbitals and single-determinant wave functions:

$$\hat{F}(i) = \hat{h}(i) + \sum_{j=1}^{n/2} [2\hat{J}_j(i) - \hat{K}_j(i)]$$

where:

$\hat{F}(i)$ is the Fock operator for the i-th electron in the system,
$\hat{h}(i)$ is the one-electron Hamiltonian for the *i-th* electron,
$n$ is the number of electrons and $n/2$ is the number of occupied orbitals in the closed-shell system,
$\hat{J}_j(i)$ is the Coulomb operator, defining the repulsive force between the *j-th* and *i-th* electrons in the system.
$\hat{K}_j(i)$ is the exchange operator, defining the quantum effect produced by exchanging two electrons.

The Coulomb operator is multiplied by two since there are two electrons in each occupied orbital. The exchange operator is not multiplied by two since it has a non-zero result only for electrons which have the same spin as the *i-th* electron. For systems with unpaired electrons there are many choices of Fock matrices.

To solve these matrices, OpenMP would be utilized, with each term depending on the basis set identified. There is a set of molecules for which the reference is obtained using the serial version of QUICK, and these molecules will be utilized as a benchmark with the new implementation. For scaling studies, the number of threads would be compared, and then the number of GPU nodes would be explored, as OpenMP is intended to be utilized in conjunction with CUDA in the end. Since the system has a variable basis set for computing the matrix, load balancing is the primary goal by introducing various conditions prior to thread distribution as each thread would take equal amount of time.