Extended abstract for poster "RHF SCF parallelization in GAMESS by Using MPI and OpenMP"

Authors: Yuri Alexeev, Graham D. Fletcher, Vitali Morozov

In the exascale era only carefully tuned applications are expected to scale to millions of cores. One of the practical solutions to achieve such scalability is to employ highly parallel methods, such as using the Fragment Molecular Orbital method (FMO)[1] in the quantum chemistry application GAMESS[2]. Other considerations include the effective use of multi-threading and the efficient use of memory. To achieve these goals for GAMESS, the following steps are required to thread Restricted Hartree-Fock (RHF) code: convert Fortran 77 code to Fortran 90 code, carry out initial code optimizations, make a thread-safe Rys quadrature integral code (by removing common blocks which may cause race conditions inside OpenMP region), and parallelize the RHF Self Consistent Field (SCF) code by using MPI and OpenMP. To accomplish these steps, we rewrote Rys integral, SCF driver, and Fock-build subroutines. We also implemented and benchmarked three OpenMP algorithms as described in this poster. All presented calculations have been run on Mira, the IBM Blue Gene/Q supercomputer located at the Argonne Leadership Computing Facility (ALCF) at Argonne National Laboratory.

The major bottleneck of an SCF procedure is the calculation of the Fock matrix. It is therefore useful to review and analyze the steps involved in the formation of Fock matrix elements. Each Fock matrix element is the sum of one- and two-electron integrals:

$$F_{ij} = h_{ij} + \sum_{kl} D_{kl}[(ij \mid kl) - \frac{1}{2}(il \mid jk)]$$

where i, j, k, l represent basis functions, **D** is the density matrix, **F** is Fock matrix, and $(ij|kl)$ is a two-electron integral. The number of two-electron integrals scale as $\sim N^4$, where N is number of basis functions. In practice, the number of integrals is reduced by symmetry of integral quartets, prescreening of integral quartets, and symmetry of the system. Because of permutation symmetry, the following integral quartets are equivalent:

$$(ij \mid kl) = (ij \mid lk) = (ji \mid kl) = (ji \mid lk) = (kl \mid ij) = (kl \mid ji) = (lk \mid ij) = (lk \mid ji)$$

The screening of the integrals is done at the shell level, so the SCF driver loops are over shells. Another potential major bottleneck is the memory requirement to store density and Fock matrices. The use of OpenMP can potentially solve this problem and make the code more efficient.

There are two major technical issues to threading GAMESS: making the code thread-safe since the use of common blocks is widespread, and the updating of the Fock matrix. Here, we present three different ways to update the Fock matrix using OpenMP threads. The algorithms have been benchmarked on a cluster of 89 water molecules for the first SCF RHF/STO-3G iteration, for no screening of integrals, and for $[ss|ss]$ integrals only. For the final work, we plan

to present benchmarking data for Rys integral code, which is also under performance optimization right now.

In the poster we present three OpenMP algorithms:
1. Atomic Fock Matrix Update. In this algorithm, we declared density and Fock matrices as shared variables. There is a potential race condition updating the Fock matrix, since different threads can write to the same Fock element. To avoid this issue, we implemented an atomic Fock update. It is the most straightforward and inefficient algorithm as it can be observed in scaling curve. The algorithm performs better than an MPI algorithm only up to 4 threads.
2. Vector Fock Matrix Update. In this algorithm, we declared density as a shared variable. We also declared two vectors as shared, which have size N_{bf}*numth, where N_{bf} is the number of basis functions and 'numth' is the number of available threads per MPI rank. This is where I and J contributions to the Fock matrix are accumulated. Here, we first realize that F_{kl} index is a "thread-safe" index since threading is done over the k index while other i and j indexes were not "thread-safe" and the corresponding Fock contributions need to be written in individual thread copy of F_i and F_j vectors. Later F_i and F_j contributions over all threads are summed up and added to the Fock matrix. The algorithm scales well and outperforms the MPI algorithm.
3. Replicated Fock Matrix Update. In this algorithm, we declared density and replicated it for each thread Fock matrix as shared variables. Here, we effectively implemented Fock private since each thread accumulated Fock contributions in its own copy of the Fock matrix. At the end all Fock contributions over all threads are summed up. This algorithm consistently outperforms the MPI algorithm, but it is not memory efficient.

Conclusions

In this work, we have applied our initial code optimization, and developed a thread-safe Rys quadrature integral code with parallelized Self Consistent Field method using OpenMP. We implemented and benchmarked three different hybrid MPI/OpenMP algorithms. The most efficient MPI/OpenMP algorithm was benchmarked against an MPI-only implementation on the IBM Blue Gene/Q supercomputer and was shown to be consistently faster by a factor of two.

Acknowledgment

This research used resources of the Argonne Leadership Computing Facility at Argonne National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under contract DE-ACO2-06CH11357.

REFERENCES

1. K. Kitaura, E. Ikeo, T. Asada, T. Nakano, M. Uebayasi, *Chem. Phys. Lett.* **313** (3–4): 701 (1999).
2. M.W. Schmidt, K.K. Baldridge, J.A. Boatz, S.T. Elbert, M.S. Gordon, J.H. Jensen, S. Koseki, N. Matsunaga, K.A. Nguyen, S. Su, T.L. Windus, M. Dupuis, J.A. Montgomery, Jr., *J. Comp. Chem.*, **14**, 1347 (1993).
3. http://www.alcf.anl.gov/mira