On Quantum Chemistry Code Adaptation for RSC PetaStream Architecture

Vladimir Mironov[1], Maria Khrenova[1], and Alexander Moskovsky[2(✉)]

[1] Chemistry Department, Lomonosov Moscow State University, Moscow, Russia
{vmironov, mkhrenova}@lcc.chem.msu.ru
[2] ZAO "RSC Technologies", Moscow, Russia
moskov@rsc-tech.ru

Abstract. Molecular simulations with quantum chemistry methods consume a large portion of CPU cycles in modern high-performance computing centers. Evolution of modern processors and HPC architectures necessitates adaptation of software to new hardware generations. The present work concentrates on the optimization of the widely used GAMESS code to Intel Xeon Phi architecture and recently devised RSC PetaStream platform. Since improvement in parallelization is required, the most frequently used Hartree-Fock and DFT methods are explored for additional parallelization options. The Xeon Phi requires vectorization that is important for electron-repulsion integrals (ERI) calculations to achieve good performance.

Keywords: Quantum chemistry · Hartree-Fock · Density functional theory · Intel xeon phi, GAMESS

1 Introduction

In 2012, Intel Many Integrated Cores (MIC) architecture [1] has been introduced as an answer to mounting challenges in building scalable and efficient high-performance computing systems. To achieve energy efficiency of computation, Intel MIC has more than 60 computational cores, each capable to execute AVX instructions. This new hardware requires new level of parallelization and vectorization from the application software for efficient performance.

Quantum chemistry algorithms were being adapted for parallel hardware for many decades. However, most popular codes "as is" don't demonstrate good performance efficiency on the Intel MIC hardware platform. In most cases, code is not vectorized, while required thread parallelism level is not achieved. For example, GAMESS(US) [2, 3] package has been parallelized for decades by now, but its code lacks vectorization and enough thread-level parallelism of important pieces of algorithm even for widely used Hartree-Fock and Density Functional Theory calculations. Intel Xeon Phi 5120D requires as many as 240 threads to be run to achieve best performance in many algorithms [4]. Attempts to run few hundred processes of GAMESS application instead of more lightweight threads overwhelm memory subsystem with dramatic performance decrease.

J.M. Kunkel and T. Ludwig (Eds.): ISC High Performance 2015, LNCS 9137, pp. 113–121, 2015.
DOI: 10.1007/978-3-319-20119-1_9

The Intel MIC set new performance per watt level for x86-compatible systems. While MIC is available as Intel Xeon Phi PCI express co-processor cards, it supports "Native" mode of application execution, where each Xeon Phi is visible to application as an independent manycore machine. The next generation of Intel MIC technology – Intel Knights Landing [5] – will be self-sufficient manycore bootable systems. Already existing RSC PetaStream architecture [6] leverages efficient co-processor-to-co-processor communication, providing realistic model of future Intel KNL supercomputers, where "native" mode of parallelization is the most natural and effective. Each node runs its own Linux-based OS image of operating system, and Linux OS is run on host. Majority of PetaStream computation power comes from Xeon Phi chips, therefore it make sense to run application on Intel Xeon Phi cards, and use the host's CPU for support and service functions; application is run on uniform field of Xeon Phi nodes – at least one MPI rank per node – compatible with "native" mode for Xeon Phi. In case of offload-like work sharing is efficient for an application and it is possible to harness both CPU and MIC nodes. RSC PetaStream system uses Intel Node Manager Technology to control and monitor node power consumption of every node, that mechanism can be used to implement flexible power energy and optimization strategies to help HPC sites save power and reduce operational costs. An example of the supercomputing system where both types of nodes co-exist in the same fabric is St. Petersburg Polytechnic University supercomputing center, with over 800 nodes on Intel Xeon E5v3 - 2697 (Haswell) share Infiniband FDR fabric with 256 nodes on Intel Xeon Phi 5120D.

The most common approaches in quantum chemistry are Hartree-Fock method (HF) and density functional theory (DFT). The major steps of these methods are construction and diagonalization of the Fock (Kohn-Sham) matrix [7]. For practically interesting systems computational power required is usually of supercomputer scale. The computational effort for the first step is dominated by the calculation of two-electron integrals corresponding to the Coulomb repulsion of electron pairs (and therefore frequently called electron repulsion integrals, ERI) and, in case of DFT, also by calculation of numerical quadrature of the exchange-correlation contribution to energy. The two-electron integral calculation has theoretical $O(N^4)$ computational complexity, where N is a number of basis functions used to characterize the system. However, many of these integrals are small enough and may be neglected. It is possible to reduce number of operations down to $O(N^{2\div3})$ using cutoffs and also some approximations, especially for very large systems. In that case a speed of Fock (or Kohn-Sham) matrix diagonalization ($O(N^3)$) significantly affects the performance of HF (DFT) method. However, for the majority of practically important molecular systems a construction of Fock (Kohn-Sham) matrix dominates overall computational cost. Also, matrix diagonalization is a pure linear algebra calculation with a great scalability, so the efficient two-electron integral code is crucial to achieve the performance in HF and DFT methods. We therefore targeted the Fock matrix two-electron contribution code to demonstrate the applicability of the Intel MIC platform to classical quantum chemistry problems.

The goal of the presented work is to enable migration of GAMESS(US) quantum chemistry code [2, 3] to novel Intel MIC hardware technology. GAMESS is widely used by the scientific community, with thousands of references in the papers each year.

We intend to minimize code modification and optimize for future-proof "native" mode of Intel Xeon Phi.

2 Basics of Hartree-Fock Method

2.1 Electron Repulsion Integrals (ERIs)

ERIs are the integrals of type:

$$I_{ijkl} = (\mathrm{i,j|k,l}) = \iint \frac{\chi_\mathrm{i}(\mathrm{r}_1)\chi_\mathrm{j}(\mathrm{r}_1)\chi_\mathrm{k}(\mathrm{r}_2)\chi_\mathrm{l}(\mathrm{r}_2)}{\mathrm{r}_2 - \mathrm{r}_1}\mathrm{dr}_1\mathrm{dr}_2 \tag{1}$$

where χ denotes basis functions, i, j, k, l – their indices, r_1, r_2 – coordinates of first and second electrons. An important property of ERIs is their eightfold permutation symmetry with respect to i, j, k, l indices. Commonly Cartesian Gaussians are used as basis functions:

$$\chi(\mathrm{r}) = (\mathrm{x} - \mathrm{A_x})^{\mathrm{a_x}}\left(\mathrm{y} - \mathrm{A_y}\right)^{\mathrm{a_y}}(\mathrm{z} - \mathrm{A_z})^{\mathrm{a_z}}\mathrm{e}^{-\alpha(\mathrm{r}-\mathrm{A})^2} \tag{2}$$

where A and α are center and exponent of basis function respectively, $a(a_x, a_y, a_z)$ – its angular momentum. They have practically important property that a product of two Gaussians is another Gaussian (see [8] for eq.). The Gaussian in form (2) is also called "primitive". Typically, linear combinations of Gaussian primitives which share the same center and angular momentum ("contracted" functions) are actually used as a basis functions. Contracted ERI are sum of integrals over their primitives:

$$(\mathrm{i,j|k,l}) = \sum\nolimits_\mathrm{a}^\mathrm{M}\sum\nolimits_\mathrm{b}^\mathrm{N}\sum\nolimits_\mathrm{c}^\mathrm{O}\sum\nolimits_\mathrm{d}^\mathrm{P}\mathrm{C_{ai}C_{bj}C_{ck}C_{dl}(ab|cd)} \tag{3}$$

where $\boldsymbol{C}$ is a matrix of contraction coefficients, M, N, O, P – degree of contraction. A set of (possibly contracted) basis functions that share the same center and same set of exponents is termed "shell". Grouping basis functions into shells reduces to some extent the number of expensive floating point operations and improves efficiency of integral screening. Primitive integrals are calculated numerically. Among the most popular approaches are McMurchie-Davidson [9], Obara-Saika [10] and Dupuis-Rys-King (DRK) [11] schemes. The effectiveness of the different schemes varies greatly for the different integral types. Quantum chemical codes often have several algorithms implemented and switch them wisely to improve performance. In this study we used only DRK integral scheme for testing purposes due to its numerical stability, relative simplicity, and uniformness for different kinds of integrals.

2.2 The Hartree-Fock Algorithm

The Hartree-Fock method is a method of finding an approximate wavefunction and energy of the model system. It is based on eigenvalue problem:

$$\mathbf{FC} = \epsilon\mathbf{SC}, \tag{4}$$

where **F** – Fock matrix, **S** – overlap matrix, **C** – matrix of molecular orbital coefficients, **ϵ** - diagonal matrix of orbital energies. Since **F** depends on **C**, the Eq. 4 has to be solved self-consistently. Matrix **F** incorporates contribution from electron-electron (V_{ee}) and electron-nuclei electrostatic interaction (V_{en}) as well as kinetic energy of electrons (T_e). It is usually represented as a sum of one-electron Hamiltonian (**h**), Coulomb (**J**) and exchange (**K**) matrices:

$$\mathbf{F} = \mathbf{h} + \mathbf{J} - \frac{1}{2}\mathbf{K} \tag{5}$$

$$h = V_{en} + T_e; J_{ij} = \sum\nolimits_{kl} D_{kl} \cdot I_{ijkl}; K_{ij} = \sum\nolimits_{kl} D_{kl} \cdot I_{ikjl} \tag{6}$$

where **D** – density matrix which is calculated from molecular orbital coefficients. Matrix **h** depends on the one-electron integrals and its computation scales quadratically depending of the system size. The Fock matrix construction requires calculation of all symmetry unique ERIs and has theoretical $O(N^4)$ complexity. It is worth noting that numerous ERIs are very small and their contribution to the Fock matrix is negligible. They could be avoided by applying screening techniques. It vastly reduces the number of ERIs required for calculation down to $O(N^{2\div3})$ depending on the geometrical size of molecular system and the nature of atomic basis set used.

Different schemes have been proposed to calculate Fock matrix. Conventional algorithm requires all ERIs to be calculated once and stored on a disk. However, it is not very efficient for the large systems due to high requirements on the amount of available disk space for the integral storage and relatively slow disk operation speed. The advantage of this method is that each ERI is calculated only once. In the alternative approach ("direct" HF) integrals are recalculated every time as needed.

3 Implementation of the Hartree-Fock Method in GAMESS

The algorithm of direct HF method implemented in GAMESS is presented on Fig. 1. The implementation of main loop over shell coefficients corresponds to the so-called "triple-sort" order [12] when up to three symmetrically unique integrals are calculated at each cycle step. The alternative is a canonical way with slightly different index order, when only one integral is calculated at each cycle step. The disadvantage of triple-sort order is decreased granularity, which may be important on highly parallel systems.

GAMESS uses MPI parallelization to split workload during ERI calculation. It is done on the ish and jsh loops implementing static and dynamic load balance. The main drawback of this implementation is a huge memory footprint on multicore architectures, because each MPI rank has its own copy of density matrix and a partial contribution to Fock matrix that scales quadratically with job size. Straightforward OpenMP implementation also inherits this drawback; however the density matrix is read-only during ERI computation cycle and could be shared between threads. The Fock matrix is constantly updated in this cycle and in simplest case it is replicated. It is

not a big problem when a large amount of memory is available. Replicated-memory MPI/OpenMP version of GAMESS was previously reported to work on Cray XT5 and further on K-computer [13]. In this algorithm each thread has its own copy of Fock matrix. Even in this case the amount of required memory reduces up to two times in comparison to original GAMESS implementation. Co-processors like MICs have large number of cores and a limited amount of on-chip memory. In this case a maximum job size is limited by the amount of available memory. A possible solution to this problem is to use distributed memory libraries like Global Arrays [14] or DDI [15]. This approach makes calculation possible even for extremely large jobs when none of these matrices could fit in a single-node memory in expense for some internode communication overhead. The distributed memory algorithms are based on the fact, that at every moment only a small amount of data from density and Fock matrix is required for the computation. Actually, only three rows of Fock matrix are updated in the innermost loop of the ERI calculation cycle. The drawback of this implementation is that interprocess communication grows, which may be quite expensive in runtime. In this study we focus on the straightforward variant of the memory problem solution.

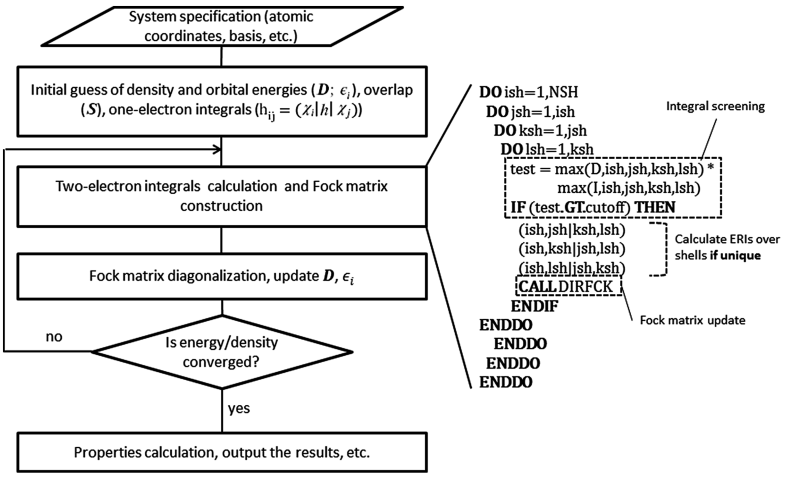

Fig. 1. Simplified algorithm of Hatree Fock implementation in GAMESS. NSH – number of shells. NSH ≤ 1000 for typical workloads.

First we tried both triple-sort and canonical way of integral ordering. They show nearly identical performance, however canonical order is slightly faster on medium-size problems due to smaller granularity. Further we always used canonical order of shells in the two-electron integral computational loop. It also has an advantage of the rectangular structure of second and third loop in nest, that could be used to improve load balance between threads.

The straightforward OpenMP version of GAMESS Fock matrix two-electron contribution shows quite a good performance on Xeon Phi (see Tables 1, 2, and 3). This implementation still has considerable memory footprint (Fock matrix is local to

```
!$omp parallel
  DO ish=NSH, 1
!$omp do schedule(dynamic,1) collapse(2)
    DO jsh=1, ish
      DO ksh=1, ish
        IF (ish.EQ.ksh) THEN
          lmax = jsh
        ELSE
          lmax = ksh
        ENDIF
        DO lsh=1, lmax
          test = max(D,ish,jsh,ksh,lsh) *
                 max(I,ish,jsh,ksh,lsh)
          IF (test.GT.cutoff) THEN
               calculate (ish,jsh|ksh,lsh)
               CALL DIRFCK
          ENDIF
        ENDDO
      ENDDO
    ENDDO
!$omp end do nowait
  ENDDO
!$omp end parallel
```

Fig. 2. Algorithm of OpenMP parallelization of the calculation of two-electron contribution to the Fock matrix in GAMESS.

each thread) but it is two times lower than for pure MPI implementation because density matrix is now shared. We observe nearly perfect parallelization when up to 60 threads per MIC are used. Further increase of number of threads per MIC improves performance only slightly. The same effect is observed on Xeon E5 CPU when more than 8 cores per socket are used.

One of the reasons of this effect is poor cache utilization when multiple threads are tied to one physical core. Indeed, the implementation of DRK algorithm of ERI calculation in GAMESS operating with large arrays of data (about L2 cache size) with nontrivial access pattern. The sizes of these arrays are set up at the compile time and depend on the maximum possible angular momentum for the basis functions. The scalability of code notably improves if we manually decrease the maximum angular momentum that code could manage from L = 7 (default in GAMESS) to L = 4. At the same time, the performance per core changes only slightly. Another reason for the scalability degradation is a poor vectorization of the ERI code in GAMESS.

It is worth noting that the scalability of code is unaltered if we consider benchmarks with similar thread/core affinity (Table 2). Therefore further improvement of single-core performance would increase overall performance as well.

The code on Fig. 2 could be also straightforwardly parallelized over the top loop in nest across MPI processes. The performance of the hybrid MPI/OpenMP version is presented in Table 3. The heaviest MPI communication task is a Fock matrix reduction that is performed only one time per HF iteration. We observe quite small ($\sim$ 1 % of execution time) synchronization and communication overhead in the case of multi-MIC run.

Table 1. Performance of the OpenMP parallelized Fock matrix two-electron contribution code for the C60 (6-31G) benchmark (KMP_AFFINITY = balanced).

Number of threads	Time of single Fock matrix build, seconds		
	Xeon E5-2690	Xeon Phi 7120D, L = 7	Xeon Phi 7120D, L = 4
1	370.8	–	–
2	195.5	–	–
4	105.5	–	–
8	55.0	815.2	790.2
16	48.5	409.6	396.4
32	–	215.2	211.4
60	–	109.7	106.6
120	–	75.6	69.0
180	–	–	61.5
240	–	79.6	59.2

Table 2. Thread affinity impact on the performance of the OpenMP parallelized Fock matrix two-electron contribution code for the C60 (6-31G) benchmark.

Number of cores used	Time of single Fock matrix build, seconds			
	1 thread/core	2 thread/core	3 thread/core	4 thread/core
16	393.1	256.4	227.8	216.1
30	212.1	137.6	123.2	117.1
60	104.9	69.3	61.3	59.2

Table 3. Performance of hybrid MPI/OpenMP parallelized Fock matrix two-electron contribution code on multiple Xeon Phi modules

Number of threads	Time of single Fock matrix build, seconds	
	C60 (6-31G), 540 b.f.	C60 (6-31G*), 900 b.f.
240 (1 MIC)	59.2	147.2
480 (2 MICs)	29.7	81.8
960 (4 MICs)	15.6	32.0
1920 (8 MICs)	8.2	25.8

3.1 Details of Benchmarks

As a benchmark systems we used fullerene molecule with two basis sets (6-31G and 6-31G*). The sizes of basis for these systems are 540 and 900 functions respectively. Xeon Phi benchmarks were conducted on RSC PetaStream platform. MIC results were compared to those of the RSC Tornado platform based on dual-socket Xeon E5-2690 server. Configurations of the test systems are summarized in Table 4.

Table 4. Configurations of the test systems

	RSC PetaStream	RSC Tornado
Host processors	1x Xeon E5-2697v2	2x Intel Xeon E5-2690
Co-processor	8x Xeon Phi 7120D	2x Xeon Phi SE10X
RAM amount/speed	128 GB DDR3R-1600	64 GB DDR3R-1600
Main board	Intel Server Board S1600JP	Intel Server Board S2600JFF
PM settings	cpufreq and PC6 enabled	EIST and Turbo enabled
Infiniband HCA	Connect-IB, 2-port	ConnectX-3 on-board
Host OS	CentOS 6.4	CentOS 6.2
MPSS	3.2.3	2.1.2
OFED version	3.5-rc3	1.5.4.1
Infiniband switch	Mellanox FDR MSX6025F. 1 hop between hosts	

4 Related Work

GAMESS [2, 3] is one of the most widely used software packages for quantum chemistry calculations. Existing parallelization in GAMESS is sophisticated [16], it has dynamic load balancing and distributed shared memory features.

GPU technology advances [17–19] created opportunity to take advantage of this new technology. NWChem code has been re-written initially for GPU [17] with CUDA technology, at its implementation on Xeon Phi [20] uses offload mode for harnessing Xeon Phi computational power. In this paper, implementation uses Xeon Phi native mode for better fitness to next generation architectures, with performance demonstration of multi-Phi. Existing GAMESS adaptation to GPU doesn't affect most widely used algorithms by computational chemists, and limited to some PCM model implementation. In more general contexts, only profiling work is reported [20]. In this respect, this paper constitutes an important contribution to the development of important software tools used by practicing researchers.

5 Conclusions

In this paper we present the design of parallelization scheme of GAMESS(US) code for quantum chemistry calculations, namely, Hartree-Fock and Density Function Theory (DFT) algorithms. Current work demonstrates the applicability of Xeon Phi coprocessors for the quantum chemistry problems. In this paper, we demonstrate scalability of the current implementation on Xeon Phi cores, as well as with multiple Xeon Phi chips running in native mode (OpenMP+MPI parallelization). Future work include more thorough performance characterization and additional vectorization of ERI calculation.

Acknowledgements. This work is supported by Intel Parallel Compute Center program. We thank Georg Zitzlsberg (Intel Corp.) and Klaus-Dieter Oertel (Intel Corp.) for valuable advices.

References

1. Goodwins, R.: Intel unveils many-core Knights platform for HPC. http://www.zdnet.com/article/intel-unveils-many-core-knights-platform-for-hpc/ (2010)
2. Schmidt, M.W., Baldridge, K.K., Boatz, J.A., Elbert, S.T., Gordon, M.S., et al.: General atomic and molecular electronic structure system. J. Comput. Chem. **14**, 1347–1363 (1993)
3. Gordon, M.S., Schmidt, M.W.: Advances in electronic structure theory: GAMESS a decade later. In: Dykstra, C., Frenking, G., Kim, K., Scuseria, G. (eds.) Theory And Applications Of Computational Chemistry: The First Forty Years, pp. 1167–1189. Elsevier, Amsterdam (2005)
4. Jeffers, J., Reinders, J.: Intel Xeon Phi Coprocessor High-Performance Programming. Morgan Kaufmann Publishers, San Francisco (2013)
5. Anthony, S.: Intel unveils 72-core x86 Knights Landing CPU for exascale supercomputing. http://www.extremetech.com/extreme/171678-intel-unveils-72-core-x86-knights-landing-cpu-for-exascale-supercomputing (2013)
6. Semin, A., Druzhinin, E., Mironov, V., Shmelev, A., Moskovsky, A.: The performance characterization of the rsc petastream module. In: 29th International Conference (ISC 2014), Leipzig, Germany, pp. 420–429 (2014)
7. Schlegel, H.B., Frisch, M.J.: Computational Bottlenecks in Molecular Orbital Calculations. Theor. Comput. Model. Org. Chem. **339**, 5–33 (1991)
8. Reza Ahmadi, G., Almlöf, J.: The Coulomb operator in a Gaussian product basis. Chem. Phys. Lett. **246**, 364–370 (1995)
9. McMurchie, L.E., Davidson, E.R.: One- and two-electron integrals over cartesian gaussian functions. J. Comput. Phys. **26**, 218–231 (1978)
10. Obara, S., Saika, A.: Efficient recursive computation of molecular integrals over Cartesian Gaussian functions. J. Chem. Phys. **84**, 3963 (1986)
11. Rys, J., Dupuis, M., King, H.F.: Computation of electron repulsion integrals using the rys quadrature method. J. Comput. Chem. **4**, 154–157 (1983)
12. Foster, I.T., Tilson, J.L., Wagner, A.F., Shepard, R.L., Harrison, R.J., et al.: Toward high-performance computational chemistry: i. scalable fock matrix construction algorithms. J. Comput. Chem. **17**, 109–123 (1996)
13. Ishimura, K., Kuramoto, K., Ikuta, Y., Hyodo, S.: MPI/OpenMP hybrid parallel algorithm for hartree − fock calculations. J. Chem. Theory Comput. **6**, 1075–1080 (2010)
14. Nieplocha, J.: Advances, applications and performance of the global arrays shared memory programming toolkit. Int. J. High Perform. Comput. Appl. **20**, 203–231 (2006)
15. Alexeev, Y., Kendall, R.A., Gordon, M.S.: The distributed data SCF. Comput. Phys. Commun. **143**, 69–82 (2002)
16. Fletcher, G.D., Schmidt, M.W., Bode, B.M., Gordon, M.S.: Distributed data interface in GAMESS. Comput. Phys. Commun. **128**, 190–200 (2000)
17. Sengottaiyan, S., Liu, F., Sosonkina, M.: A GPU support for large scale quantum chemistry applications. In: The 2012 International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA 2012), Las Vegas, Nevada, USA (2012)
18. Ufimtsev, I.S., Martínez, T.J.: Quantum chemistry on graphical processing units. 1. strategies for two-electron integral evaluation. J. Chem. Theory Comput. **4**, 222–231 (2008)
19. Ufimtsev, I.S., Martinez, T.J.: Quantum chemistry on graphical processing units. 2. direct self-consistent-field implementation. J. Chem. Theory Comput. **5**, 1004–1015 (2009)
20. Aprà, E., Klemm, M., Kowalski, K.: Efficient implementation of many-body quantum chemical methods on the intel® xeon phi™ coprocessor. In: Proceedings of the International Conference for High Performance Computing, Networking, Storage and AnalysisPiscataway, NJ, USA, pp. 674–684. IEEE Press (2014)