ARGONNE **LEADERSHIP COMPUTING** FACILITY

MPI+OpenMP Paralelization of Density Functional Theory Method in GAMESS

Vladimir Mironov[1], Yuri Alexeev[2], Dmitri G. Fedorov[3]
[1]Department of Chemistry, Lomonosov Moscow State University; [2]Computational Science Division, Argonne National Laboratory;
[3]National Institute of Advanced Industrial Science and Technology

Abstract

In this work, Density Functional Theory (DFT) method is parallelized with MPI-OpenMP in quantum chemistry package GAMESS. It has been implemented in both regular and Fragment Molecular Orbital (FMO) based DFT codes. The scalability of the FMO-DFT code was demonstrated on Cray XC 40 Theta supercomputer. We demonstrated excellent scalability of the code up 2,048 Intel Xeon Phi nodes (131,072 cores). Moreover, the developed DFT code is about twice faster than the original code because of our new grid integration algorithm.

GAMESS (General Atomic and Molecular Electronic Structure System

- ❑ A general purpose comprehensive *ab initio* quantum chemistry package
- ❑ Maintained by the research group of Prof. Mark Gordon at Iowa State University (http://www.msg.ameslab.gov/gamess).
- ✓ Enables most major quantum mechanical methods (Hartree-Fock, Møller-Plesset perturbation theory, coupled cluster, multiconfiguration self consistent field, configuration interaction, density functional theory).
- ✓ Ported to many computer architectures.
- ✓ Free and widely used on everything from laptops to supercomputers.
- ✓ Highly scalable, including many distributed data algorithms.
- × More than a million lines of legacy Fortran 77 code, with an associated parallelization library comprising 15,000 lines.
- × Parallelization library does not support multithreading

Fragment Molecular Orbital (FMO) Method

- ❑ In FMO, a molecular system is divided into fragments and the total properties, such as the energy or its gradient, is calculated from those fragments and (in FMO2) their pairs, computed in the embedding potential. FMO treats the electrostatics for the whole system while the exchange-repulsion and charge transfer are only accounted for at the local level of fragment pairs.
- ❑ The total energy can be written as:

$$E = \sum_{I}^{N_{frag}} E_I + \sum_{I}^{N_{frag}} \sum_{J<I}^{N_{frag}} \left(E_{IJ} - E_I - E_J\right)$$

where the monomer (I) [FMO1], dimer (IJ) [FMO2] energies can be obtained using any standard QM method

- ❑ Current applications:
  - geometry optimization (~2,000 atoms)
  - ab initio level single point energies (~20,000 atoms)
  - pair interaction analysis (~20,000 atoms)
  - drug design, ligand docking, chemical reactions in solution etc
- ❑ Massively parallel FMO utilizes the Generalized Distributed Data Interface (GDDI):
  - With the molecule divided into fragments, each fragment or pair is computed on a distinct subset, or group, of the available processors
  - Each fragment/pair is then run in parallel in each group
  - This provides two levels of parallelization, greatly increasing the parallel efficiency

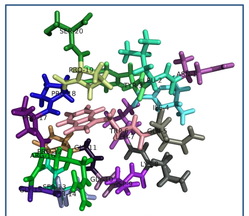


Figure 2. Example of FMO fragmentation scheme applied to a 20 amino-acid residue peptide (1L2Y)

Density functional theory (DFT) method

- One of the most popular quantum chemistry methods
- High precision results for relatively low computational cost
- Very similar to the Hartree-Fock method

Legacy DFT implementation in GAMESS:

1. Inefficient algorithms of numerical grid calculation, which are slow and result in too large grid
2. In-house algorithms for Fock-matrix update
3. For the nuclear gradients the unnecessary grid weight contribution is always computed
4. C-style access to multidimensional array with grid data

DFT algorithm modernization strategy:

- Introduce modern DFT algorithms
- Use optimized BLAS subroutines if possible
- Optimize access to the data

```
Set up numerical grid (r_i, ω_i)
repeat                                              ▷ Begin SCF loop
    Compute Coulomb and optionally exact exchange terms
    for each (r_i, ω_i) do
        Compute electron density ρ(r_i)
        (Optional, GGA) compute density gradient ∇ρ(r_i)
        (Optional, mGGA) compute kinetic energy density τ(r_i)
        Compute f_i^XC(ρ, ∇ρ, τ)
        Update the Fock matrix using
            F_mn += ω_i f_i^XC ϕ_m(r_i) ϕ_n(r_i)        ▷ rank-2 update
    end for
    Diagonalize the Fock matrix, update ρ and ϵ
until |D − D^old| < conv or |ϵ − ϵ^old| < conv
```

Figure 1. Generic algorithm of DFT method

Goals of the study

- ❑ Develop multithreaded DFT code in GAMESS to reduce following drawbacks of the MPI implementation:
  - Excessive memory utilization on massively parallel CPU architectures for legacy codes like GAMESS.
  - Only half of GAMESS processors are doing computation, other are used as data-servers.
- ❑ Bring recent advances of the DFT method to GAMESS users
- ❑ Create a framework for DFT method development in GAMESS

FMO algorithm in GAMESS

Basic steps of FMO (FMO2) calculation:

1. Compute one-electron (core) Hamiltonian of each fragment
2. Construct initial guess wavefunction of each I fragment
3. Monomer self consistent charge (SCC) field calculations. Compute the wavefunction of fragments in a field of other fragments. The latter is described by the embedding electrostatic potential (ESP). This step loops until all wavefunctions converge
4. Dimer calculations. They are done once at the end
5. Aggregate the results and compute properties

Problem of FMO-DFT calculation:

- Computing XC contribution to the Fock matrix is a major bottleneck for small chemical systems, like FMO fragments.

```
Calculate one-electron Hamiltonian of each fragment
Guess initial density matrix for each fragment D_i
    ▷ Monomer calculation:
repeat
    for i = 1, NumFrag do
        GDDI load balancing
        Compute embedding ESP quasi-Fock matrix
        Compute wavefunction of the fragment i, update
        energy and density matrix of fragment i (ϵ_i, D_i)
        Compute fragment properties
        Broadcast results (energy, density, charges)
    end for
until max(|D_i^old − D_i|) < conv
    ▷ Dimer calculation:
for i = 1, NumFrag do
    for j = 1, i − 1 do
        GDDI load balancing
        Compute embedding ESP of the dimer
        Compute wavefuction, get energy (ϵ_ij) and
        density matrix(D_ij) of the fragment pair ij
    end for
end for
Gather results of fragment pairs calculation
Calculate whole system properties, print results
```

Figure 3. Algorithm of FMO (FMO2) method

Hardware description

	Theta (ALCF)	Intel Xeon single node
Processor	Intel Xeon Phi 7230, 64 cores, 1.3 GHz	2x Intel Xeon Gold 6254
Memory per node	16 GB MCDRAM + 192 GB DDR4	1.5 TB DDR4
Number of nodes	3,624	1
Interconnect type	Aries interconnect with Dragonfly topology	-
Compiler	Intel Parallel Studio XE 2018.2	Intel Parallel Studio XE 2018.2

New DFT implementation

- Store grid as a list of clusters of points:
  - Discard clusters of points whose contribution to the Fock matrix is less than a threshold
  - Clustering allows to use BLAS level 3 operations when updating Fock matrix
- Whenever possible skip computing and storing Becke's factors to quadrature weights
- MPI+OpenMP parallelization both on grid construction and XC integration steps
- Static/dynamic load balancing for MPI processes and dynamic load balancing for OpenMP threads

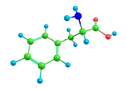
```
Zero XC contribution to Fock matrix: F(:,:) = 0
!$omp parallel do schedule(dynamic) reduction(F)
for m = 1, N_clusters, MPI_ChunkSize do
    MPI static/dynamic load balancing
    Get data of all points in Q_m
    for each point p in Q_m do
        Compute AO densities ρ_i^AO
        (Opt.) Compute ∇ρ_i^AO
        (Opt.) Compute ∇(∇ρ_i^AO)
        if all ρ_i^AO < ρ_thr then
            Mark point p for removal
        end if
    end for
    if cluster Q_m is empty then
        Mark cluster m for removal
    end if
    Compute ρ_i^MO for remaining points using BLAS
    (Opt.) Compute ∇ρ_i^MO
    Compute XC values on a grid
    Update Fock matrix using BLAS
end for
if grid was changed then
    Synchronize grid among MPI processes
end if
```

Figure 4. Integration of XC functional

```
Set up partitioning function σ(μ)
    ▷ Set up grid point clusters:
for each atom do
    Compute spherical grid S for an atom
    Compute radial grid R for an atom, split on intervals R_i
    ▷ Atomic "sphere of influence":
    Compute r_min : ∀r < r_min, σ(μ(r)) = 1
    ▷ Construct clusters of points for an atom:
    for each radial interval R_i do
        Divide points of spherical grid to spherical sectors S_j
        Store clusters of grid points Q_ij = R_i × S_j in memory
    end for
end for
    ▷ Process grid point clusters:
!$omp parallel do schedule(dynamic)
for m = 1, N_clusters, MPI_ChunkSize do
    MPI static load balancing
    if r_max(Q_m) < r_min then
        Mark cluster as non-modified
    else
        for each spherical direction sph in cluster do
            for each radii value rad in cluster do
    ▷ Process point p(rad, sph):
                Compute w_p^0 = w_rad w_sph
                if p in sphere of influence of another atom then
                    w_p = 0
                    exit radii loop
                end if
                Compute Becke's factors f(σ) for point p
                and all other atoms
                Compute w_p from w_0 and f(σ)
                if w_p = 0 then
                    exit radii loop
                end if
            end for
        end for
    end if
    if all w_p = 0 then
        Drop cluster Q_m
    end if
end for
Synchronize grid among MPI processes
```

Figure 5. Molecular grid construction

Benchmark description



DFT setup:

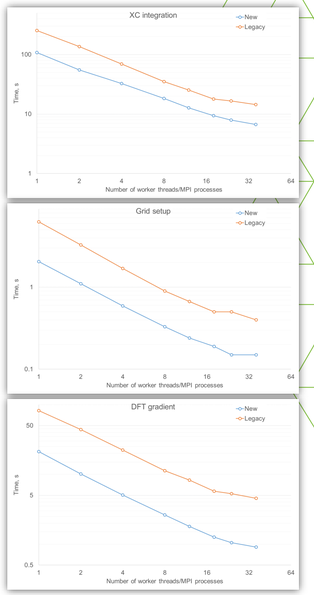
- Phenylalanine aminoacid:
  - Moderate-size aminoacid
- B3LYP/6-31G* energy + gradient calculation
  - Precise DFT grid (96 radial × 1202 angular grid points per atom)
- Run on single node to compare the performance of both codes

FMO-DFT setup:

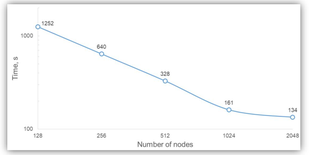
- Spherical water droplet with 30 Å radius
- FMO2-B3LYP/6-31G* energy calculation
- Large-scale benchmarking on Theta supercomputer

Benchmark results

Single node benchmarks



FMO-DFT benchmarks on Theta



Conclusions

In this work, Density Functional Theory (DFT) method has been parallelized with MPI-OpenMP in quantum chemistry package GAMESS. The parallel performance has been measured on a single Intel Xeon Phi node and supercomputer Theta (Cray XC40 based on Intel Xeon Phi).The developed hybrid MPI+OpenMP code scales well up to 2,048 Intel Xeon Phi nodes. Moreover, we also rewrote DFT grid integration code, resulting in about 2x performance improvement compared to the original code.

Supercomputing 2019 | November 17-22 2019, Denver, Colorado (USA)

Acknowledgments: This research used resources of the Argonne Leadership Computing at Argonne National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under contract DE-ACO2-06CH11357. This research was supported by the Exascale Computing Project (17-SC-20-SC), a collaborative effort of the U.S. Department of Energy Office of Science and the National Nuclear Security Administration. VM thanks Russian Foundation for Basic Research (project #19-03-00043). Authors thank Intel Corporation for providing access to computational resources.