ARGONNE **LEADERSHIP COMPUTING** FACILITY

Heuristic Dynamic Load-Balancing Algorithm Applied to the Fragment Molecular Orbital Method

Yuri Alexeev[1], Prasanna Balaprakash[1,2]

[1]Leadership Computing Facility, Argonne National Laboratory (USA); [2]Mathematics and Computer Science Division, Argonne National Laboratory (USA)

Abstract

The load balancing for large-scale systems is an important NP-hard difficulty problem. We propose a heuristic dynamic load-balancing algorithm (HDLB), employing the Covariance Matrix Adaptation Evolution Strategy, a state-of-the-art heuristic algorithm, as an alternative to the default dynamic load balancing (DLB) and previously developed heuristic static load-balancing algorithms (HSLB). The problem of allocating CPU cores to tasks is formulated as an integer nonlinear optimization problem, which is solved by using an optimization solver. On 16,384 cores of Blue Gene/Q, we achieved an excellent performance of HDLB compared to the default load balancer for an execution of the fragment molecular orbital method applied to model protein system quantum-mechanically. HDLB is shown to outperform default load balancing by at least a factor of 2, thus motivating the use of this approach on other coarse-grained applications.

Supercomputer Challenge

ALCF supercomputer Mira Blue Gene/Q architecture:
- 1,024 nodes per rack
- 16 cores/node
- 1.6 GHz processor
- 16 GB of memory/node

Mira has a total of
- 48 racks (768,000 cores)
- 768 terabytes of RAM
- peak performance of 10 petaflops
- 384 IO nodes
- 240 GB/s, 35 PB storage

ALCF PC cluster Cooley Intel Haswell architecture
- 126 compute nodes
- 12 CPU cores and one NVIDIA Tesla K80 dual-GPU card per node
- 2.4 GHz processor
- 384 GB of memory/node

A challenge for quantum chemistry codes such as GAMESS is to efficiently use the enormous number of cores which require well parallelized and scalable algorithms. For example, in this work a number of parallelization of algorithms were attempted for efficient parallelization of MEP code.

Mira and Cooley used for DOE-run INCITE and ALCC projects

Innovative and Novel Computational Impact on Theory and Experiment (INCITE) and Argonne Leadership Computing Challenge (ALCC) programs aim to accelerate scientific discoveries and technological innovations by awarding, on a competitive basis, time on supercomputers to researchers with large-scale, computationally intensive projects that address “grand challenges” in science and engineering. 2015 INCITE Allocations at ALCF: 3.57 billion core-hours on Mira.

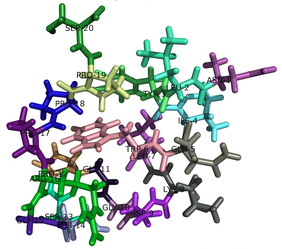
GAMESS

(General Atomic and Molecular Electronic Structure System)

- *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 most major computer architectures.
- Free and widely used on everything from laptops to supercomputers.
- About a million lines of code, with an associated parallelization library comprising 15,000 lines.
- Highly scalable, including many distributed data algorithms.

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.



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

The total energy can be written as: $E = \sum_{I}^{N} E_I + \sum_{I>J}^{N} (E_{IJ} - E_I - E_J)$

where the monomer (I), 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.

FMO code structure and explanation of load-balancing setup

```
Do
   For i=1, Monomers                          <- Monomer SCC
      Dynamic load balancing
      Compute integrals and solve F
      Broadcast densities/charges
   End
While not converged
monomer densities
For i=1, Dimers                               <- Dimers
      Dynamic load balancing
      Compute integrals and solve F
End
```

Each phase has its own set of MPI subgroups for computational efficiency

There are two stages in FMO calculations: monomer self-consistent charge (SCC) field calculations and dimer calculations at the end, which are done once. In this work, we are concerned only with load balancing in monomer stage. For load-balancing tests, we chose the protein which consists of 76 fragments. We have created 76 GDDI groups (i.e., number of MPI subgroups) equal to the number of fragments. In the pseudo code on the left in red, it is shown where assignment of MPI ranks to each MPI subgroup is done.

FMO load-balancing algorithms

There are a few ways to load balance computation of the fragments in monomer stage of FMO calculations. In the default dynamic load-balancing and static load-balancing algorithms, all MPI subgroups are roughly same size. The algorithm tries to balance assignment of fragment to different subgroups. In our developed HSLB and HDLB algorithms, we keep assignment of fragments fixed (one fragment per MPI subgroup) and vary in size of each MPI subgroup for the optimal time to solution across all MPI subgroups. It is an NP-hard optimization problem. HSLB and HDLB vary in the way this optimization problem is solved.

Heuristic static load-balancing algorithm

A key feature of HSLB method is the formulation of a mixed-integer nonlinear optimization (MINLP) problem to model the allocation of processing cores to tasks. There are four steps: gather data (run GAMESS 3-5 times on 1k to 16k nodes), fit data, solve MINLP, and execute FMO. Refer to our SC12 publication for more details. The problem with this approach is that at least 3-5 calculations of the whole system are required on a large variety of nodes, which may not be possible (for example, we typically vary number of nodes from 1k to 16k nodes). In addition, MINLP requires AMPL program which is a proprietary non-portable commercial code.

Heuristic dynamic load-balancing algorithm

- We adopt the Covariance Matrix Adaptation Evolution Strategy (CMA-ES), a state-of-the-art heuristic evolutionary optimization algorithm for dynamic load balancing.
- Like any other evolutionary algorithm, CMA-ES performs the following steps: i) Initialize parent population; ii) Generate offspring population by using selection, recombination, mutation operators; iii) Select new parent population by combining the parent and offspring population.
- CMA-ES uses a distribution for mutation which is generated according to a covariance matrix C; this corresponds to learning a second order model of the objective function.
- Consequently, mutations can adapt to the shape of the objective function landscape and convergence to the optimum can be increased significantly.
- Statistics collected over the generations are used to control covariance matrix and other algorithmic parameters.
- In order to handle the integer parameters, the decision point is rounded to the nearest integer only during the objective function evaluation.

CMA-ES is used iteratively as follows:

1. Gather load-balancing data from previous iteration.
2. Fit a simple linear speedup model for each fragment (when a fragment ***f*** runs for ***s*** sec with ***n*** processes, the product of ***n*** and ***s*** gives the time for single process).
3. Use the linear speedup models for all fragments as an objective function for CMA-ES.
4. Deploy CMA-ES to find the best load-balancing allocation and reallocate the resources.

We have compared performance of HDLB against DLB and HSLB, see Figure 2. Like HSLB, HDLB on average outperforms default DLB by an order of two. But HDLB is much easier to use, and we plan to implement HDLB directly in the GAMESS. There is no need to run preliminary 3-5 calculations. As shown on Figure 1, HDLB converges quickly on the first iteration. There is, in fact, no need to run any extra calculations. On Figure 3, we have shown that HDLB has the same efficiency as HSLB. It is worth mentioning that the difference between the maximum and minimum time will decrease as the number of nodes increases. This difference exists because of the constraints in the setup.

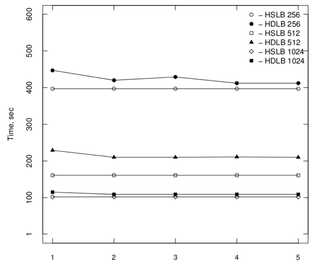


Figure 1. Comparison of HDLB performance vs HSLB for different number of nodes varying from 256 to 1024.

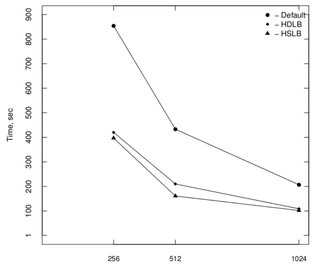


Figure 2. Comparison of scaling HDLB with default dynamic load balancing and HSLB.

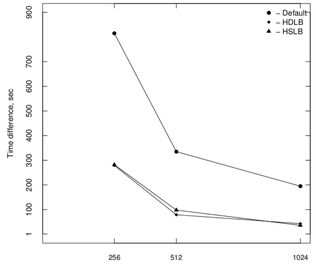
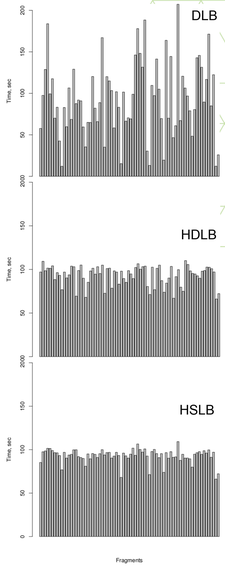


Figure 3. Time difference between maximum and minimum time for solution for HDLB, DLB, and HSLB.



Conclusions

We have shown that the present HDLB method is twice as fast as the default DLB method and has comparable performance to HSLB. HSLB algorithm provides an optimal solution, but it is hard to use, especially in the community code like GAMESS because MINLP uses a commercial package AMPL. On the other hand, HDLB utilizes an effective and simple algorithm to achieve the same goal. However, there is evidently room for improvement. We plan to continue our work trying different objective functions and also expanding HDLB approach to FMO dimer calculations.

Argonne NATIONAL LABORATORY

U.S. DEPARTMENT OF ENERGY

Argonne National is a U.S. Department of Energy laboratory managed by UChicago Argonne, LLC.

Acknowledgments 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-AC02-06CH11357.