Task-based parallel computation of the density matrix in quantum-based molecular dynamics using graph partitioning

Purnima Ghale
University of Illinois, Urbana Champaign
Los Alamos National Laboratory
Email: ghale2@lanl.gov

Georg Hahn
Imperial College London
Los Alamos National Laboratory
Email: ghahn@lanl.gov

Matthew Kroonblawd
University of Missouri, Columbia
Los Alamos National Laboratory
Email: mkroonblawd@lanl.gov

Sergio Pino
University of Delaware
Los Alamos National Laboratory
Email: sergiop@lanl.gov

Vivek Sardeshmukh
University of Iowa, Iowa City
Los Alamos National Laboratory
Email: viveks@lanl.gov

Guangjie Shi
University of Georgia
Los Alamos National Laboratory
Email: sgjerry@lanl.gov

***Abstract*—Quantum molecular dynamics (QMD) simulations are highly accurate, but they are computationally expensive due to the calculation of the ground-state electronic density matrix $\mathbf{P}$ via an $O(N^3)$ diagonalization. Second-order spectral projection (SP2) is an efficient $O(N)$ alternative to obtain $\mathbf{P}$ from a Hamiltonian matrix $\mathbf{H}$. This poster presents a data-parallel version of the SP2 algorithm that uses an undirected graph representation of the matrix $\mathbf{P}$ to divide the computation into smaller independent partitions. These partitions can give rise to undesirable load imbalances in standard *MPI/OpenMP*-based implementations, as they are often of unequal sizes. The load-balancing problem is addressed by using task-based programming models to schedule parallel computations during runtime. We present *CnC* and *Charm++* implementations that can be integrated into existing QMD codes. Our approach is applied to QMD simulations of representative biological protein systems with more than $10,000$ atoms, exceeding size limitations of diagonalization by more than an order of magnitude.**

1. Introduction

Quantum molecular dynamics (QMD) simulations are a highly accurate tool to predict material properties, with potential applications in targeted pharmaceuticals, safety and reliability testing, semiconductor synthesis and fabrication.

In QMD simulations a system of atoms evolves over a series of time steps according to forces computed from the Hamiltonian matrix $\mathbf{H}$ and its corresponding electronic density matrix $\mathbf{P}$. In practice, $\mathbf{P}$ is computed from $\mathbf{H}$ up to 10^5 times or more for a typical QMD simulation. $\mathbf{P}$ is traditionally obtained from an $O(N^3)$ diagonalization of $\mathbf{H}$, thus limiting the number of simulated atoms N to around 10^3 [3], [4].

Second-order spectral projection (SP2) is an $O(N)$ alternative for systems with sparse $\mathbf{H}$ and $\mathbf{P}$ which replaces diagonalization with a polynomial expansion of $\mathbf{H}$ using a recursive series of generalized matrix-matrix multiplications. Dense and sparse linear algebra representations of the SP2 algorithm have been implemented for both shared and distributed memory CPU architectures, as well as for GPU accelerators [3], [4], [8].

In this work, we develop a data-parallel version of the SP2 algorithm based on an undirected graph representation of $\mathbf{P}$. Since traditional graph partitioning schemes do not account for the physics underlying our application, new partitioning heuristics are developed. The proposed algorithms are implemented in a parallel fashion and can be included in standard QMD codes.

2. The proposed parallelization of SP2

We start with a molecular system we wish to investigate using QMD, such as the simulation of proteins solvated in water. The Hamiltonian matrix $\mathbf{H}$ describes the behavior of electrons in the system, and $\mathbf{P}$ represents its essential features. When the matrix $\mathbf{P}$ is interpreted as an undirected graph, each non-zero element in $\mathbf{P}$ is an edge in the graph, and each atomic orbital is a vertex. Partitioning the graph corresponding to $\mathbf{P}$ allows us to divide $\mathbf{H}$ and thus subsequent SP2 computations into smaller independent submatrices (subproblems) which are then solved with a data parallel approach.

In order to obtain partitions of $\mathbf{P}$ we extend usual partitioning schemes based on minimizing the edge-cut between partitions (such as *METIS* [5], *hMETIS* [6] or *KaHIP* [10]). For the SP2 algorithm to work accurately and to obtain accurate physics for inner as well as for boundary atomic orbitals, it is important to not only consider the elements in each sub-Hamiltonian but also all its nearest neighbors. Matrix multiplications of order $O(N^3)$ constitute the main computational effort of the SP2 algorithm, hence the effort for each partition is of order $(c_i + h_i)^3$, where c_i is the

number of inner *core* vertices and h_i is the number of nearest neighbors (the *halo* of the partition). To summarize, we are trying to divide the graph representation of $\mathbf{P}$ into q partitions with the aim to minimize $\sum_{i=1}^{q}(c_i + h_i)^3$ (1).

To this end, we found that standard packages such as the ones aforementioned are reasonably well suited if used with the right objective function, such as the minimization of the overall communication volume. Most importantly, improved results can be obtained by using our own scheme based on simulated annealing [7] tailored to the objective function (1) which we employ to further optimize partitions computed by standard packages.

3. Asynchronous Task Based Programming

The proposed parallelization obtained from graph partitioning can give rise to undesirable load imbalances as the partitions are generally not of equal size. In bulk synchronous parallel (BSP) implementations, the programmer has to explicitly specify how data is distributed to ensure that needed elements are available when a task is being executed, thus making them susceptible to load imbalances. Asynchronous task-based programming models form an alternative to BSP as they allow us to mitigate the load-balancing problems by efficiently scheduling parallel computations during runtime. In these task-based approaches, a computation is expressed in terms of tasks and their dependencies, and this information is used by the runtime to schedule the tasks based on available resources [1], [2], [9]. We develop two versions of our graph partitioned parallel SP2 using Intel Concurrent Collections *CnC* and *Charm++*.

Our data parallel implementations are applied to QMD simulations of representative biological protein systems with more than $10,000$ atoms, thus exceeding size limitations of diagonalization by more than an order of magnitude. We present results on how our approach scales with both an increasing number of partitions/cores and with molecules of increasing size, carried out for both the *CnC* as well as the *Charm++* implementation.

When assessing our *CnC* and *Charm++* implementations for a varying number of partitions (subproblems), the advantage of dividing the matrix into smaller subproblems becomes evident in the rapid decrease in runtime. This rapid decrease is very evident initially as the number of partitions is increased. Even when the matrices are over-decomposed into subgraphs beyond this range, the runtime flattens out for a smaller test system composed of water molecules and continues to improve for a larger protein system.

We also compare the strong scaling behavior of both the *CnC* and the *Charm++* implementations for the same water and protein systems with 60, 120, and 1200 partitions per core. We see that the *CnC* scalability is limited to less than 64 cores, while *Charm++* continues to improve.

4. Summary and poster design

In this interdisciplinary project we demonstrate the benefits of employing graph partitioning to divide a large computational problem into data-parallel subproblems for a targeted application in QMD simulations. The problem under investigation and background information on QMD is presented in the corners of our poster.

The core of our approach consists of employing standard graph partitioning algorithms as well as new heuristic approaches tailored to our specific flavor of the graph partitioning problem with the aim to parallelize the workload of computing the density matrix needed for QMD simulations. Load imbalances which possibly arise in BSP models due to the unequal sizes of data-parallel partitions are mitigated by using asynchronous task based programming models. These steps are arranged as blocks around a circular arrow in clockwise direction in the center of the poster. We carry out simulation studies of the performance of the data-parallel approach, and the results of those simulations are presented in the lower left corner of the poster.

Acknowledgments

The authors would like to thank their mentors at Los Alamos National Laboratory: Ben Bergen, Nick Bock, Marc Cawkwell, Hristo Djidjev, Christoph Junghans, Sue Mniszewski, Christian Negre, Anders Niklasson, Robert Pavel, and Ping Yang.

References

[1] B. Acun, A. Gupta, N. Jain, A. Langer, H. Menon, E. Mikida, X. Ni, M. Robson, Y. Sun, E. Totoni, L. Wesolowski, and L. Kale. Parallel programming with migratable objects: Charm++ in practice. *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 647–658, 2014.

[2] Z. Budimlić, M. Burke, V. Cavé, K. Knobe, G. Lowney, R. Newton, J. Palsberg, D. Peixotto, V. Sarkar, F. Schlimbach, and S. Taşırlar. Concurrent collections. *Scientific Programming*, 18(3-4):203–217, 2010.

[3] M.J. Cawkwell, E.J. Sanville, S.M. Mniszewski, and A.M.N. Niklasson. Computing the density matrix in electronic structure theory on graphics processing units. *J. Chem. Theory and Comput.*, 8(11):4094–4101, 2012.

[4] M.J. Cawkwell, M.A. Wood, A.M.N. Niklasson, and S.M. Mniszewski. Computation of the density matrix in electronic structure theory in parallel on multiple graphics processing units. *J. Chem. Theory and Comput.*, 10(12):5391–5396, 2014.

[5] G. Karypis and V. Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM J. Sci. Comput.*, 20(1):359–392, 1999.

[6] G. Karypis and V. Kumar. Multilevel k-way hypergraph partitioning. *VLSI Design (Very-large-scale Integration)*, 11(3):285–300, 2000.

[7] S. Kirkpatrick, C.D. Gelatt Jr, and M.P. Vecchi. Optimization by simulated annealing. *Science*, 200(4598):671–680, 1983.

[8] S.M. Mniszewski, M.J. Cawkwell, M.E. Wall, J. Mohd-Yusof, N. Bock, Germann T.C., and A.M.N. Niklasson. Efficient parallel linear scaling construction of the density matrix for born-oppenheimer molecular dynamics. *J. Chem. Theory and Comput.*, 2015.

[9] E.H. Rubensson and E. Rudberg. Chunks and tasks: A programming model for parallelization of dynamic algorithms. *Parallel Computing*, 40(7):328–343, 2014.

[10] Peter Sanders and Christian Schulz. Think Locally, Act Globally: Highly Balanced Graph Partitioning. In *Proceedings of the 12th International Symposium on Experimental Algorithms (SEA'13)*, volume 7933 of *LNCS*, pages 164–175. Springer, 2013.