A Task-based Execution Model for Coupled Cluster Methods

Anthony Danalis, University of Tennessee

Computational chemistry, aiming to simulate non-trivial physical systems, imposes such high demands on the performance of software and hardware, that it comprises one of the driving forces of High Performance Computing. In particular, many-body systems, such as those simulated by the Coupled Cluster (CC) methods of the Quantum Chemistry package NWChem [4] are both computationally intensive and of interest to the Computational Chemistry community.

Despite the need for high performance, harnessing large fractions of the processing power of modern large scale computing platforms has become increasingly difficult over the last few decades. This is true due to both the increasing scale and the increasing complexity and heterogeneity of modern (and projected future) platforms. We believe that dataflow-driven task-based programming models may be the only viable way for achieving computation at scale, especially on distributed heterogeneous architectures.

The Parallel Runtime Scheduling and Execution Control (*PaRSEC*) framework [3] is a task-based dataflow-driven runtime that enables high performance computing at scale. *PaRSEC* enables task-based applications to execute on distributed memory heterogeneous machines, and provides sophisticated communication and task scheduling engines that hide the complexity of supercomputers from the application developer, while maximizing the achievable performance. The main difference with other task engines is the way tasks and their dependencies are represented, which enables *PaRSEC* to employ a unique way of discovering and processing the DAG. Namely, *PaRSEC* uses a symbolic Parameterized Task Graph (PTG) to represent the tasks and their dependencies on other tasks.

The PTG is a problem-size-independent representation that allows for immediate inspection of a task's neighborhood, regardless of the location of the task within the DAG. This contrasts all other task scheduling systems, which discover the tasks and their dependencies at run-time (through the execution of skeleton programs), and therefore cannot process a future task that has not yet been discovered.

The PTG supports extremely scalable distributed execution and provides *PaRSEC* with the capability to tolerate memory and synchronization latencies, as well as communication jitter, while exposing the maximum degree of parallelism. *DPLASMA* [2] (the distributed dense linear algebra library we have developed over *PaRSEC*), offers a testament to the scalability offered by *PaRSEC*. As an example, *DPLASMA* has been successfully used on systems with thousands of CPU cores and systems with multiple GPUs per node, and has demonstrated significant performance gains over the state-of-the-art, i.e., vendor provided libraries such as the *Cray Scientific Libraries package (libSci)*.

However, utilizing a task scheduling system such as PaRSEC to execute the Coupled Cluster code of NWChem is not trivial. When an application is structured as tasks with well defined inputs and outputs — i.e., pure functions — and execution spaces that are defined by affine loops, then PaRSEC provides a mostly automated path from source code to the PTG representation. However, the CC code — generated by the Tensor Contraction Engine (TCE) [1] — is neither organized in pure tasks, nor is the control flow affine. For example, the code contains branches whose predicates depend on program data. This makes the code not only non-affine, but statically undecidable. However, the program data that controls the behavior of these branches is set only once, upon the initialization stages of the code, and does not change during the execution. Capitalizing on this fact, we have created *PaRSEC* versions of CC code subroutines. This endeavor involved creating special functions that perform the same data lookups as a CC subroutine at run-time, but before the subroutine executes. These special functions populate custom meta-data structures, but do not execute any of the computationally intensive kernels found in the original CC subroutine — i.e., `GEMM`, `SORT`, etc. Due to the existence oe this meta-data, when converting a CC subroutine into the PTG form, we can include run-time lookups into the meta-data structures. This enables *PaRSEC* to take full control of the task and communication scheduling, while preserving the semantics of the original CC code.

This effort has demonstrated the feasibility of converting TCE generated codes into a PTG form that can be scheduled by *PaRSEC*. In future work, we plan to pursue the automation of a large scale conversion from TCE generated code into a PTG form. Further, we aim to achieve tighter integration of *PaRSEC* and TCE in order to enable NWChem to harness large scale systems, as well as accelerators such as GPUs and Intel Xeon Phi.

References

[1] G. Baumgartner, A. Auer, D.E. Bernholdt, A. Bibireata, V. Choppella, D. Cociorva, X. Gao, R.J. Harrison, S. Hirata, S. Krishnamoorthy, S. Krishnan, C. Lam, M. Nooijen, R.M. Pitzer, J. Ramanujam, P. Sadayappan, and A. Sibiryakov. Synthesis of High-Performance Parallel Programs for a Class of Ab Initio Quantum Chemistry Models. *Proceedings of the IEEE*, 93(2):276–292, 2005.

[2] G. Bosilca, A. Bouteiller, A. Danalis, M. Faverge, A. Haidar, T. Herault, J. Kurzak, J. Langou, P. Lemarinier, H. Ltaeif, P. Luszczek, A. YarKhan, and J. Dongarra. Flexible development of dense linear algebra algorithms on massively parallel architectures with DPLASMA. In *Proceedings of the Workshops of the 25th IEEE International Symposium on Parallel and Distributed Processing (IPDPSW 2011)*, pages 1432–1441. IEEE, 16-20 May 2011.

[3] G. Bosilca, A. Bouteiller, A. Danalis, T. Herault, P. Lemarinier, and J. Dongarra. DAGuE: A generic distributed DAG engine for high performance computing. *Parallel Computing*, 38(12):37 – 51, 2012. Extensions for Next-Generation Parallel Programming Models.

[4] M. Valiev, E.J. Bylaska, N. Govind, K. Kowalski, T.P. Straatsma, H.J.J. van Dam, D. Wang, J. Nieplocha, E. Apra, T.L. Windus, and W.A. de Jong. NWChem: a comprehensive and scalable open-source solution for large scale molecular simulations. *Comput. Phys. Commun.*, 181(1477), 2010.