Report of a Workshop on

Parallelization of Coupled Cluster Methods

Held February 23 and 24, 2008 at the King and Prince Resort on St. Simon's Island, Georgia, as part of the 48th Sanibel Symposium

Submitted to the National Science Foundation and Department of Energy
May 8, 2008

Submitted on behalf of the Participants by:
Rodney J. Bartlett and Erik Deumens
bartlett@qtp.ufl.edu
deumens@qtp.ufl.edu

DISCLAIMER

The workshop described in this report was supported by the National Science Foundation under grant number CHE-0803118 and by the Department of Energy under grant number DE-FG02-08ER15934. Any opinions, findings, or conclusions are those of the authors and do not necessarily reflect the views of the National Science Foundation or the Department of Energy.

EXECUTIVE SUMMARY

The evolution of modern computer technology towards multi-core processors and special purpose graphics processor chips and the installation at national centers of petaflops computer systems presents both great opportunities and great challenges for the computational chemistry community in general and the community of developers of coupled cluster methods in particular. The latter methods are the acknowledged benchmark for problems with around 20 atoms and about 100 active electrons, but with the recent developments toward the petascale it is anticipated that such benchmark methods will begin to be applied to molecules with 200 atoms and active 1000 electrons. As such the potential for creating definitive results for some of the most important problems in chemistry will emerge. With minor modifications, CC is the method of choice for the description of nuclear and atomic physics, exemplified by the workshop at the Institute for Nuclear Theory **Atomic, Chemical, and Nuclear Developments in Coupled Cluster Methods** (INT-08-2a) organized June 23 - July 25, 2008 by David Dean, Rodney Bartlett, Walter Johnson, Achim Schwenk.

CONCLUSIONS AND RECOMMENDATIONS

There was strong consensus among workshop participants about the following issues and ideas.

1. **Communication (between researchers)**
    a. Keep the communication open. It is essential to stimulate collaboration within the computational chemistry community and with the computer science community. A particular challenge is educating computer science students about the application domains because these skills are not the most marketable for the typical positions computer scientists hold. It is also important to collaborate with the NSF centers to get effective access to the resources.
    b. This workshop should be repeated on a regular basis to coordinate this software and skill development effort and to optimize the communication.
    c. Collaborations with computer scientists and applied mathematicians are needed. These are hard to establish because computer scientists and applied mathematicians are working on their own problems.
    d. The challenge posed by the petascale computers to software development goes beyond the capacity of small, single PI research groups to handle. A software development collaboration model must be found to join efforts of many groups in an effective way. The open source model is one example. Can it work for this task?
    e. Support for software development in this effort is a critical and essential investment by NSF and DOE.
    f. QTP will maintain a table of benchmark data on the web to allow easy communication and comparison of results obtained with new algorithms and new implementations. The link to the table with benchmark results is http://www.qtp.ufl.edu/PCCworkshop/PCCbenchmarks.html

2. **Tools for developers**
    a. A better understanding of tools is needed and this group should provide input for tool developers regarding
        i. Debuggers
        ii. Performance measurement instrumentation and analysis tools
        iii. Specialized tools:
            1. Global Array Toolkit [1]
            2. Disk Resident Arrays [2]
            3. Distributed Data Interface [3]
            4. Super Instruction Architecture [4]
            5. Charm++ http://charm.cs.uiuc.edu/ for C++
            6. GASnet http://gasnet.cs.berkeley.edu/ a low-level, language independent communication system for languages like UPC, Titanium and Co-Array Fortran.
        iv. Higher level tools, such as automatic code generation by the Tensor Contraction Engine
    b. Adaptable, flexible, extensible programming models defined by APIs and possibly by using domain-specific languages should be developed to enhance programmer productivity.
    c. The routine and quick-turnaround availability of 1,000 processors is crucial for development, debugging and tuning petascale applications. The identification of a developers group with special access, such as the NSF the solicitation Petascale Computing Resource Allocation (PRAC) seems a good idea and may need to be expanded and repeated.
3. **Scientific focus:** This community should formulate grand challenge problems that have broad appeal and can be easily and clearly communicated, for example, the full ab-initio treatment of cuprate superconductors with high critical temperature is such an open problem that is clearly still a challenge.
4. **Parallelism**
    a. Hierarchical parallelism should be exploited as part of the strategy to effectively use petascale computers
        i. Run one 10,000-processor job
        ii. Run many 1,000-processor jobs
        iii. Create groups inside a 10,000-processor job that perform different tasks coordinated to contribute to the big job
    b. Think about the workflow of how to do the calculation outside the paradigm of a single, large job.
    c. Fault tolerance will become an essential part of every petascale application. A good start would be a fault tolerant implementation of MPI with automated process rescheduling and message rerouting The MPI standard may have to be extended to include calls API to manage fault tolerance.
5. **Education and Training:** Students need to be educated both in chemistry and in computer science to be able to contribute to the solution of the petascale problem. A suggested curriculum should include:

a. Programming (most current programming courses focus on Java and C++, even if other languages like Fortran have advantages for high-performance kernels, they is easy to learn for an experienced programmer) using OpenMP, POSIX Threads, MPI.
b. Cache hierarchy is crucial to understand; it is taught in computer architecture classes, not programming classes.
c. Numerical method courses are rare in computer science departments and the one taught in mathematics departments have a more academic flavor.
d. Use of on-line and web-based material should be encouraged because it can effectively reach a geographically dispersed group. Good resources are available from NCSA and PSC.

6. **Vision for computational chemistry petascale software:** The following capabilities are desirable
    a. One-particle theory: HF, TDHF, DFT, TDDFT
    b. Many-body theory:
        i. MBPT(2) (MP2)
        ii. CCSD, EOM-CCSD
        iii. CCSD(T), EOM-CCSD(T)
        iv. CCSDT, EOM-CCSDT
        v. R12-CC
        vi. State-specific multi-reference CC
    c. Connection: ab-initio DFT, seamless transition between DFT and wave function theory.
    d. Density matrix renormalization group approaches.

    All require analytical gradient capability to do chemistry, and would benefit from analytic Hessians and higher analytical derivatives. Much of the future will involve serious considerations of anharmonic effects, for example, while non-linear optics and two-dimensional spectroscopy need such higher derivative information, including its time-dependence.

PARALLEL EFFICIENCY

Thom Dunning summarized the kind of systems that need to be considered when thinking about programming for and using petascale super computers:

1. The Cray XT6: 27,888 quadcore processor chips provide 111,552 processors, it has 223 TB of RAM, and 45,000 hard disk drives with a 240 GB/s transfer rate to disk, it takes up 3,400 square feet of floor space, and uses 7.5 MW of electrical power which calls for 6,000 tons of cooling.
2. The IBM PERCS "Blue Waters" to be installed at NCSA by July 2011: more than 200,000 cores, more than 800 TB of RAM, more than 10 PB of disk storage, it will take up 5,000 square feet of floor space and will be water cooled.

High latency will become a problem for communicating small blocks. The petascale computers will not have networks that are full fat trees. That would be too expensive in cabling. The network topology therefore becomes an important issue and must be considered when designing algorithms.

Heterogeneous computers will be unavoidable. This leads to the requirement of carefully considering the data movement between different parts of memory and how they can be accessed by the different processors at the correct time in the algorithm. Data flow models will be crucial as well as techniques for task scheduling. Do we need dynamic load balancing to maintain performance?

The use of globally shared disks will run into the problem of finding the correct balance for number of file system servers for a system. This may depend strongly on applications so that there is no good choice that can be made at the system administrator level and will serve a majority of users and applications. A new protocol may have to be developed to support a dynamic number of file system servers depending on the load from the application. Maybe there is a need for "regional file systems", intermediate between "local" and "global"?

Complex systems will have failures within the time frame of (almost) every application. Fault tolerance therefore becomes a crucial concern. The MPI standard and current MPI implementations assume that all tasks remain active throughout the run. If one task fails, the entire parallel computation must abort. A fault tolerant implementation is available from a company called EverGreen. It is not clear whether their approach scales to petascale computers. A new MPI standard must address the capability for a running parallel program to manage itself. Is it feasible to checkpoint a petascale application? Or does one checkpoint parts of the set of all running tasks?

To build efficient implementations of Coupled Cluster methods on petascale computers with their inherent complexity will require involving computer scientists. However, teaching domain science to computer science students has been found to be very hard. This learning often requires a lot of background that computer science students do not have. Acquiring that background takes a significant amount of time that many students do not wish to invest. To them the domain science material may be interesting but may not be valued at all in their future workplace environment.

An Open Source Project requires core management and staff. The original NWChem project had on around 8 people in the core group at PNNL plus a larger community of collaborators. One may be available to do something in computer science like Linux and Apache without explicit funding, but that is not possible in chemistry. If chemistry is to be able to take advantage of the coming generation of petascale computers (and even to take full advantage of current terascale computers), NSF and DOE must commit adequate funding for a core effort in the development of petascale chemistry applications. A formalized but flexible collaborative workflow paradigm to engage computer and domain scientists in common projects is not yet available, but this would foreseeably accelerate progress in scaling coupled-cluster codes and algorithms to the petascale.

Summary

1. Hierarchical parallelism.
2. Collaboration with computer scientists is essential.
3. New theories, methods and algorithms must be considered and developed, in particular ones that scale linearly.
4. Global file systems will be increasingly important.

PARALLEL COUPLED CLUSTER

Large Coupled Cluster problems require a large set of T amplitudes, too large to hold in one node. Integrals can be recomputed, but the T amplitudes must be communicated. To keep code simple, this will require efficient one-sided communication.

Because petascale networks will not be full fat trees, the network topology must be considered when designing algorithms and choosing the data distribution. Data locality will become almost necessary.

Coupled Cluster theory is in some sense easy to parallelize, because it is dominated by matrix operations. However, focusing on the matrix algebra alone has been shown to be insufficient to make Coupled Cluster methods scale well.

The design of parallel Coupled Cluster implementations could, in addition to considering computing individual contractions in parallel, consider computing several diagrams in parallel. The total number of tasks can be divided into groups and each groups computes a number of diagrams. The code executed could be significantly different for different diagrams, taking into account the different scaling and computational requirements of each class of diagrams.

The complexity of Coupled Cluster equations and the complexity of petascale computers can only be managed by increasing the use of automatic code generator tools, such as TCE (tensor contraction engine). Automatic code generation can contribute in three ways:

1. Validation of theoretical ideas. By generating correct code quickly a new theoretical idea can be tested for scientific validity and usefulness, even if the implementation by the code generator is not efficient.
2. Reference for optimized implementation. The code generated can be used as a starting point for performance analysis and tuning with other tools or by hand. The results obtained with the automatically generated code can be used as reference for hand optimized implementation.
3. Programming tool. Automatic code generators do not have to generate complete, ready-to-compile programs. They can be useful for a programmer by generating error-free sections of code that implement complex formulas.

The Coupled Cluster problem in a very large basis is ill-conditioned.

1. One way to control ill-conditioning is to use more accurate arithmetic. The standard accuracy of 64 bit floating point hardware will not be sufficient for all calculations involving so many floating point numbers. Maybe “interval arithmetic” as promoted by Sun can be used to investigate and control the seriousness of cumulative rounding errors?
2. Another way is to formulate a modified method or algorithm that avoids or controls the instability.

Summary

1. Focus on data locality will be crucial to achieve petascale performance.
2. Automatic code generation is unavoidable.

BENCHMARKS

A committee should define a list of benchmark problems. A benchmark problem is a precisely defined problem that can be performed by anyone interested. Benchmarks are useful for several reasons

1. The table of results and timings of benchmark problems provides any researcher who makes a new implementation of an existing method or implements a new method or algorithm with a reference as to how many resources (processors, RAM, disk) the calculation requires and how long the calculation should take.
2. The table of hardware platforms and timings of benchmark problems assists researchers in the design of a computer system when they are considering purchasing and building a computer for their laboratory or organization.
3. The table of benchmark problems, hardware configurations, software used, and timings helps users to select the appropriate software for a given calculation and to estimate how many resources they should request and how long they should expect the calculation to take

Traditionally the performance was measured in floating point operations. However, petascale Coupled Cluster calculations will also move a large amount of data and this can have serious impact on the total performance visible to the user, which is wall-clock time to completion.

The website with the benchmark tables can be found at:
http://www.qtp.ufl.edu/PCCwokshop/PCCbenchmarks.html

Summary
Availability of a well-maintained set of benchmark problems is critical for ensuring stable progress in a complex collaboration as the one required to scale coupled cluster methods to petascale computers.

PETASCALE ALGORITHMS

How do we scale algorithms to tens of thousands of cores? Will they be completely new algorithms? Or, can existing algorithms be adapted?

What problems are we trying to solve?

1. Electronic, vibrational and rotational spectra.
2. Energies and response properties.
3. Geometries of ground states and transition states and reaction paths.

Are Coupled Cluster methods still the methods of choice? Competing high-accuracy methods are quantum Monte-Carlo (QMC) and density matrix renormalization group (DMRG). QMC cannot be used for time-dependent studies at this time. But for the foreseeable future, only CC theory, plus its EOM-CC treatment of excited states, ionized, electron attached states and response properties, is able to describe essentially all properties of interest to chemistry. Simpler methods like DFT and TDDFT have prospects for calculating such properties in very large systems, but they cannot provide benchmarks since they do not necessarily converge to a physically realizable approximate solution as must ab initio CC methods..

What algorithms will be needed? Most likely a single algorithm will not do. Algorithms may need to be chosen after the problem size and other parameters are known and after the petascale computer and its architecture are known to get a reasonably optimal computation.

Developers of petascale software may have to implement several algorithms, each emphasizing a different alternative from the list below:

1. Store or recomputed. Recomputation will be more favored on modern processors that are many orders of magnitude faster than memory, but there still are cases where storing intermediate data is better in an absolute sense or in the sense that the total application wall-clock time to completion is shorter.
2. Using sparsity. Examples are using a three-body approximation or Cholesky decomposition of the two-electron integrals.
3. R12 integrals exploit sparsity and a Coupled Cluster singes and doubles (CCSD) result with a triple-zeta basis with R12-technology is equivalent to one with a quintuple-zeta basis without R12-technology.
4. Linear scaling methods rely on maximizing the benefit of localization in basis sets, interactions, and correlation.

In the area of materials simulations and computations, quantum chemists can help in providing methods for treating electron correlations in solids and especially, experience in applying methods such as coupled cluster theory. It may not ever be possible to fully implement coupled cluster theory in a first principles code for solids, but it is worth applying coupled cluster theory at a lower level, say at a coupled cluster singles and doubles level, over a limited number of particle states. Lessons learned from these calculations on solids may prompt significant improvements in mean field methods such

as hybrid density functional methods, where the Hartree-Fock exchange is 'screened' to a large extent.

Also outside quantum chemistry, CC methods for nuclei have been shown to provide some of the best results available.

In order to make a significant impact on experiment, it is important to be able to treat systems which are metallic or nearly so. Ideally one would like to be able to predict electronic structures close to metal insulator transitions. And these are particularly challenging to any method, first principles or model Hamiltonian alike.

Summary
Multiple algorithms may have to be considered and implemented in software with the flexibility to choose the best one at run time when the problem and the computer are known.

REFERENCES

1. I. Nieplocha, R.J. Harrison, and R. J. Littlefield. Global arrays: A nonuniform memory access programming model for high-performance computers. In Proceedings of Supercomputing 1994, page 340, Washington D.C., 1994. IEEE Computer Society Press.
2. Sriram Krishnamoorthy, Gerald Baumgartner, Chi-Chung Lam, Jarek Nieplocha, and P. Sadayappan. Layout transformation support for the disk resident arrays framework. J. Supercomput., 36(2):153-170, 2006.
3. Graham D. Fletcher, Micheal W. Schmidt, Brent M. Bode, and Mark S. Gordon. The distributed data interface in GAMESS. Comp. Phys. Comm., 128:190-200, 2000.
4. V. Lotrich, N. Flocke, M. Ponton, A. D. Yau,_ A. Perera, E. Deumens, and R. J. Bartlett, Parallel Implementation of Electronic Structure Energy, Gradient and Hessian Calculations, J. Chem. Phys., 2008, accepted.

APPENDIX I: WORKSHOP FORMAT

The workshop was held on February 23 and 24, 2008 at the King and Prince Resort on St. Simon's Island, Georgia, as part of the 48th Sanibel Symposium. The workshop was started with two focused plenary speaker sessions on Saturday that were part of the Sanibel Symposium program and a talk to open the afternoon discussion session.

1. Thom Duning, Jr., University of Illinois, "Meeting the Challenge of Petascale Computing."
2. Bert de Jong, Pacific, Northwest National Laboratory, "Pushing the Scientific Envelope on Large Computing."
3. Peter Pulay, University of Arkansas, "An efficient Parallel Implementation of the External Exchange Operator and Perturbative Triples Contributions to Coupled Cluster Energies."
4. Karol Kowalski, Pacific Northwest National Laboratory, "New Coupled Cluster Approaches for Modeling Molecular Properties in Various Environments."
5. Jozef Noga, Cornelius University, "Towards a more Accurate and more Efficient Coupled Cluster Implementation in the Bratislava Group."
6. Victor F. Lotrich, ACES QC and University of Florida, "ACES III: Parallel Implementation of Electronic Energy, Gradient, and Hessian Calculations."
7. Ryan Olson, Cray Research, "Parallel CC and Petaflops applications."

Saturday afternoon from 2 pm until 7 pm and Sunday afternoon from 1 pm until 5 pm, the workshop participants had discussions organized as follows. Then the workshop had discussions organized in four panel sessions. The panel members had been asked to prepare a 10 minute statement to introduce the topic of the panel discussion.

Panel 1 *Parallel efficiency*
Topic: Impact of communication protocols and memory use, MPI-1, MPI-2 and one-sided communication, shared memory and OpenMP, input-output to disk storage from parallel programs.
Panel members: Curt Janssen, Tomasz Janowski**,** Bert de Jong, Thom Dunning

Panel 2 *Parallel Coupled Cluster*
Topic: Issues related to parallel implementation of algorithms to solve coupled-cluster equations, what is similar to and different from algorithms used in other large-scale parallel software, e.g. molecular dynamics, finite element analysis, weather simulation.
Panel members: Josef Noga, Karol Kowalski, John Watts

Panel 3 *Benchmarks*
Topic: Can we define a set of problems that everyone can run with their software on their hardware and report to the community? Similar to the Linpack benchmark in linear algebra, we need some molecules and some properties, energy, gradient, with a list of methods. Maybe also a molecule like a polymer that can be made larger in a defined way to provide more work as the number of processors is increased
Panel members: Peter Pulay, Sudhakar Parmidighantam, P. (Saday) Sadayappan

Panel 4 *Petascale algorithms*
Topic: What algorithms can we use to scale CC methods to 1,000, 10,000 and 100,000 processors? Do we need all new algorithms? How do we measure performance? How do we debug?
Panel members: Ed Valeev. So Hirata. Shawn Brown, Beverly Sanders, Charles Patterson

Conclusions and recommendations
Formulation of conclusions and recommendations of the workshop.

In addition to the formal discussion, there was ample opportunity Saturday and Sunday for many personal discussions. There was a reception Sunday evening to close the workshop. Several participants stayed at the Sanibel Symposium longer and had further discussions. This report tries to capture as many of the issues and ideas discussed as possible.

APPENDIX II: LIST OF PARTICIPANTS

Rod Bartlett
Shawn T. Brown
Christopher Chang
Ernest Davidson
Bert de Jong
Erik Deumens
Thom H. Dunning Jr.
Norbert Flocke
Laszlo Fusti-Molinar
Steven Gwaltney
Thomas M. Henderson
So Hirata
Thomas F. Hughes
Tomasz Janowski
Curtis Janssen
Bogumil Jeziorski
Karol Kowalski
Tomasz Kus
Victor F. Lotrich
Ann Melnichuk
Monika A. Musial
Jozef Noga
Ryan Olson
Sudhakar Parmidighantam
Charles H. Patterson
Michal Pitonak
Mark Ponton
Peter Pulay
P. (Saday) Sadayappan
Beverly A. Sanders
Peter Schmidt
Andrew G. Taube
Edward F. Valeev
Robert Vergenz
Prakash Verma
Thomas Watson
John D. Watts
Charles A. Weatherford
Kizashi Yamaguchi
Shuske Yamanaka