Computational chemistry at the petascale: Are we there yet?

Journal of Physics: Conference Series **180** (2009) 012027 doi:10.1088/1742-6596/180/1/012027

Computational chemistry at the petascale: are we there yet?

E Aprà, R J Harrison, W A Shelton, V Tipparaju and A Vázquez-Mayagoitia

Computer Science and Mathematics Division, Oak Ridge National Laboratory, Oak Ridge, TN

E-mail: aprae@ornl.gov

Abstract.

We have run computational chemistry calculations approaching the Petascale level of performance ($\sim$ 0.5 PFlops). We used the Coupled Cluster CCSD(T) module of the computational chemistry code NWChem to evaluate accurate energetics of water clusters on a 1.4 PFlops Cray XT5 computer.

1. Introduction

The field of electronic structure is struggling to get efficient parallel implementation on Petascale class hardware. One notable exception has been the achievement of Qbox, a planewave pseudopotential electronic structure code that obtained a performance of 207 TFlops on a BlueGene/L computer [1].

Qbox makes use of the message-passing MPI library for parallelization. Instead, NWChem makes use of the Global Arrays library; this allows the software developer to reach a high level of abstraction and, at the same time, to use one-sided communication to efficiently exploit the network hardware. In the remainder of the paper, we will discuss recent benchmarks and scientific results obtained with NWChem on a parallel computer whose theoretical peak performance is in excess of 1 PFlops.

2. Parallelization approaches

Multi-level parallelization has been successfully used by several authors ([2], [3] or [4]) to achieve good parallel performance at large scale. This parallelization approach is applied to simulation methods where it is possible to subdivide the calculation in a large number of independent tasks; therefore the parallelization issue remain restricted at the level of each task that typically use only a few hundred of computational cores.

As an example of this multilevel parallelization approach, our research group is currently implementing an algorithm for the calculation Raman vibrational intensities by density functional theory. Raman spectra intensities depend on the changes of the molecular polarizability due the normal mode vibrations. Specially, the frequency dependent polarizability evaluation requires demanding quantities of computational resources, and its derivatives are too complex to be evaluated analytically. Our implementation of the Raman intensities divides the computation uniformly in groups of processors by numerical derivatives, each group having

a computational task whose size is consistent with the scalability limit of solution of the polarizability problem [5].

However, in the rest of the paper we will describe an computational scheme that does not resort to subdivide the computational load at multiple level, but implementation a parallelization approach that uses all the available cores together, in their entirety.
Before moving to the description of the scientific algorithm, we will briefly describe the main aspect of the parallelization framework used in NWChem.

3. The Global Arrays library on the Cray XT5

The primary component of the Global Arrays library is the distributed arrays layer. This forms an **abstraction layer** that greatly simplifies the effort of the developer of the scientific application by isolating most of the intricacies involved in parallelize an application that uses dense matrices. For example, this layer does all the necessary translation from the users shared memory styled access to the actual distributed array. It has several components that it uses: a Message Passing library, the ARMCI one-sided communication library and a memory allocator (MA library). The Distributed Arrays layer is mostly implemented on top of the ARMCI one-sided communication library.

A message passing library (MPI) is used in the Global Arrays for process management and for some collective communication . For the most of the NWChem modules (including the CCSD(T) module that we discuss below), the amount of MPI usage is minimal, most of the communication is via ARMCI one-sided communication library using the non-contiguous one-sided communication interfaces it provides.

The ARMCI communication library has interfaces for contiguous, strided and vector communication along with Read-Modify-Write operations. All of the Global Array one-sided calls are translated into ARMCI communication calls. ARMCI library, underneath, uses the fastest possible mechanism to transmit data. It uses shared memory with-in the node and, on the Cray XT5 system, uses Portals library for inter-node communication. The Global Arrays library is optimized to overlap intra-node data transfers in shared memory and inter-node data transfers in the network using non-blocking calls. On the Cray XT5, in addition to regular MPI tasks, ARMCI spawns a thread to assist with communication (particularly Read-Modify-Write and non-contiguous data transfer operations). Each of the cores on the octacore XT5 runs an MPI task or, as an alternative, one can sacrifice one core for the sever thread (discussed below) and use the remaining seven cores for the computing pool.

4. CCSD(T) implementation in NWChem

Coupled-cluster (CC) is a numerical many-body technique that studies the effect of electron correlation on the electronic structure of molecular systems. CCSD(T) is one of several CC methods that estimates the effect of electron correlations by considering single, double and triple excitations; single and double excitations are fully computed with a self-consistent approach, while the contribution of the triple excitations is computed pertubatively [6]. Here we describe the CCSD(T) module of the NWChem library, its usage of the Global Arrays and discuss how Global Arrays library is structured on the Cray XT5.

Valence only coupled cluster CCSD(T) [6] calculations have been called the "gold standard of quantum chemistry" [7] for their chemical accuracy in determining molecular energetics. The computational cost of CCSD(T) calculations formally scales as the seventh power of the number of basis functions (N^7), hence this is a method that could effectively utilize large supercomputers. A few parallel implementations of CCSD(T) are available [8–12], however, most of them have been designed either for clusters or for moderate scale parallelism.

In this work we have substantially modified and enhanced the original parallel implementation of CCSD(T) in NWChem of Kobayashi and Rendell [8] that was designed to effectively

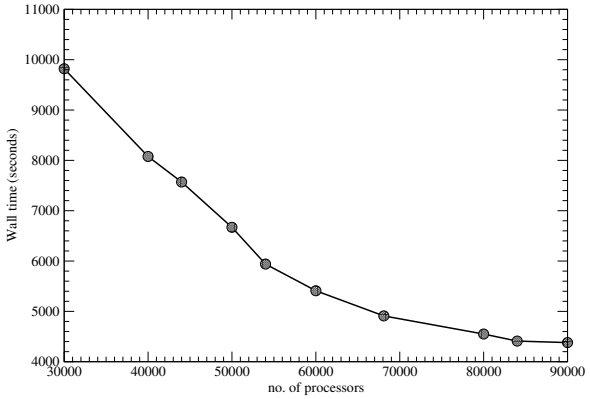

Figure 1. Walltime for the CCSD(T) calculation of total energy of $(H_2O)_{18}$ as a function of the number of processors

utilize massively parallel processors and to make minimal use of I/O resources. The focus of the performance numbers reported here is the perturbative triples correction implemented in NWChem following the "*aijkbc* algorithm" of Rendell and coworkers [13] that makes no use of I/O by storing intermediate quantities (two-electron integrals and coupled-cluster wave function amplitudes) in the global memory managed by Global Arrays.

The floating-point intensive kernel of this algorithm is a series of calls to the BLAS DGEMM [14] matrix multiplication routine. Several data parallel and one-sided Global Arrays operations are used in the CCSD(T) implementation of NWChem, most notable of these are the `ga_get()` and `ga_acc()` calls that are used to get/accumulate sections of distributed array.

Our revisions to this algorithm emphasized increased locality to reduce communication and implemented a more careful tiling of intermediates to reduce memory consumption and increase parallelism and load balance. The dynamically load balanced algorithm explicitly considers three levels of the memory hierarchy. It proceeds by tiling the full computation so that intermediate results fit in available global memory, then tiling the nested loops so that data associated with each task fits into local memory. A process access a global shared counter to determine the next task, moves data from global to local memory using the `ga_get()` operation, computes, and accumulates results into the result using the `ga_acc()` operation. The local computation employs the BLAS DGEMM [14] matrix multiplication routine to optimize for the local memory hierarchy.

5. Results

We reported performance number by using as base the parallel implementation of CCSD(T) in NWChem of Kobayashi and Rendell [8] that was designed to effectively utilize massively parallel processors and to make minimal use of I/O resources.

What distinguishes the benchmark numbers reported here is the unprecedented scale of the calculations and floating-point performance achieved. We run a series of benchmark with the 5.1 version of NWChem [15] using the water cluster $(H_2O)_{18}$ and a modified cc-pvtz basis set [16] for a total of 918 Gaussian basis functions; wall-time of the CCSD(T) runs as a function of processor number are reported in Figure 1. The last datapoint at 90000 processor reached a sustained 64-bit floating-point performance of 358 TFLOP/s.

All eight cores of the octacore Cray XT5 node were used in the performance results reported here. In order to alleviate memory usage and improved load-balancing, we had to modify the distribution of the larger Global Arrays used in this algorithm. The effect of last two distribution

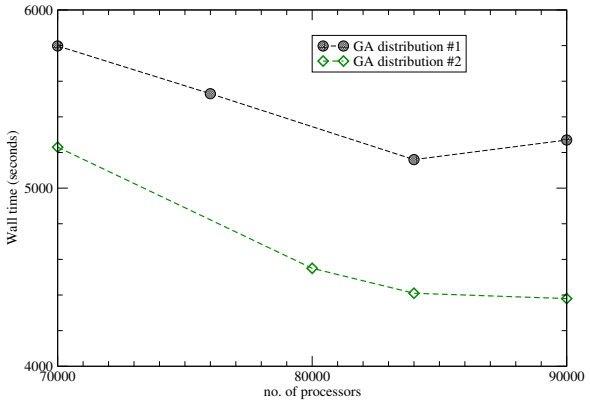

Figure 2. $(H_2O)_{18}$ benchmark: effect of different `ga_create` distributions

approaches we developed are shown in the plot[1] of Figure 2.

We modified the 5.1 NWChem source code in two main aspects by moving communications outside of inner loops (sometime at the additional cost of more use of local memory) and by modifying the distribution of the largest Global Arrays used in the triples kernel of the CCSD(T) method. The first more obvious distribution we attempted (labeled as distribution #1 in the plot) allowed us to scale at 7000 processors, but it could not scale beyond 80000 processors. This first distribution was already an improvement compare to the previous one since it drastically cut down the memory requirements (the previous one would have required 2GB or more of memory, clearly not an option on a system that has only 2GB of RAM on each core). In the next step, we refined the distribution by having the large Global Arrays being distributed over as many processors as possible; the finer granularity of this second distribution improved load-balancing and allowed us to reach better scaling beyond 80000 processors.

A further benchmark run, on the even larger cluster $(H_2O)_{20}$, we ran a coupled-cluster calculation on the ORNL's Jaguar petaflop computer that used over 100 TB of memory for a sustained performance of 487 TFLOP/s (double precision) in the triples sections on 96,000 processors, lasting for 2 hours. This floating-point performance corresponds to 55% of theoretical aggregate performance of the 96000 2.3GHz AMD Opteron cores. The number of basis functions for this run is equal to 1020 and the number of correlated orbitals is equal to 80. While the 487 TFLOP/s refers just to the triples section of the calculation, if we consider the run from beginning to end (including I/O operations such as reading the 21GB amplitudes file plus the initial step of transforming the Atomic Orbitals into Molecular orbitals), the floating point results is of 444 TFLOP/s. In figure 3 we reported the scaling curve of this benchmark.

64-bit floating precision is a necessary requirement for this kind of calculations because of the error propagation in the computation of the gaussian integrals being accumulated and of the numerical instability of high accuracy gaussian basis functions given used.

As far as scientific results are concerned, we present here (table 1) some preliminary numbers that show the relative stability of twenty water clusters in two different conformations. The binding energies computed in this study are of great value for describing the the properties of water at the molecular level.

[1] The data point at 70000 processors in the upper curve of the right plot was obtained by interpolating between the measured values at 50000 and 76000 processors

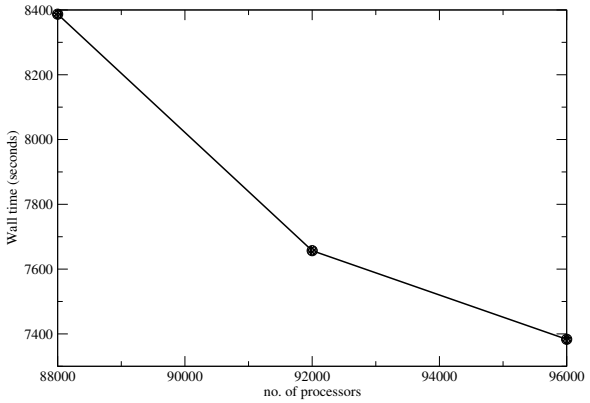

Figure 3. Walltime for the CCSD(T) calculation of total energy of $(H_2O)_{20}$ as a function of the number of processors

6. Conclusion

A unique aspect of this work is the use of the one-sided Global Arrays programming model that enabled us to scale a single tightly-coupled calculation to reach a performance approaching the petascale level. We project performance approaching 1 PFlops on larger water clusters on an upgraded computer systems.

Acknowledgments

This work was supported by the Department of Energy, offices of Basic Energy Science and Advanced Scientific Computing Research as part of the SciDAC program, and was also in part sponsored by the Laboratory Directed Research and Development Program of Oak Ridge National Laboratory (ORNL), managed by UT-Battelle, LLC for the U. S. Department of Energy under contract DE-AC05-00OR22725 with Oak Ridge National Laboratory. This research used resources of the National Center for Computational Sciences at Oak Ridge National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC05-00OR22725.

[1] Gygi F, Draeger E W, Schulz M, de Supinski B R, Gunnels J A, Austel V, Sexton J C, Franchetti F, Kral S, Ueberhuber C W and Lorenz J 2006 *SC '06: Proceedings of the 2006 ACM/IEEE conference on Supercomputing* (New York, NY, USA: ACM) p 45 ISBN 0-7695-2700-0

[2] Fedorov D G, Olson R M, Kitaura K, Gordon M S and Koseki S 2004 *Journal of Computational Chemistry* **25** 872–880

[3] Windus T L, Kathmann S M and Crosby L D 2008 *Journal of Physics: Conference Series* **125** 012017 (10pp) URL `http://stacks.iop.org/1742-6596/125/012017`

[4] Wang L W, Lee B, Shan H, Zhao Z, Meza J, Strohmaier E and Bailey D H 2008 *SC '08: Proceedings of*

	Edge-sharing		Dodecahedron	
	E^{tot}	B.E.	E^{tot}	B.E.
SCF	-1521.371595	152.87	-1521.363070	147.52
MP2	-1526.467996	251.02	-1526.444136	236.05
CCSD	-1526.539945	228.14	-1526.518107	214.43
CCSD(T)	-1526.709594	245.10	-1526.684946	229.64

Table 1. Total Energies (E^{tot}) in hartrees and Binding energies (B.E.) in Kcal/mol for the edge-sharing pentagonal prism and the dodecahedron $(H_2O)_{20}$ structures

the 2008 ACM/IEEE conference on Supercomputing (Piscataway, NJ, USA: IEEE Press) pp 1–10 ISBN 978-1-4244-2835-9

[5] Johnson B G and Florin J 1995 *Chemical Physics Letters* **247** 120 – 125 ISSN 0009-2614 URL http://www.sciencedirect.com/science/article/B6TFN-3TJ5JRM-50/2/080969b3cc20c5a690f9a439f97ef378

[6] Raghavachari K, Trucks G W, Pople J A and Head-Gordon M 1989 *Chemical Physics Letters* **157** 479 – 483 ISSN 0009-2614 URL http://www.sciencedirect.com/science/article/B6TFN-44F1NYB-58/2/dc7ab0fbf4f6f7beddd4f5e9f7494ac6

[7] Dunning T H, Peterson K A, Woon D E and Wilson A K 1999 *American Conference on Theoretical Chemistry* unpublished

[8] Kobayashi R and Rendell A P 1997 *Chemical Physics Letters* **265** 1 – 11 ISSN 0009-2614

[9] Hirata S 2003 *The Journal of Physical Chemistry A* **107** 9887–9897 (*Preprint* http://pubs.acs.org/doi/pdf/10.1021/jp034596z) URL http://pubs.acs.org/doi/abs/10.1021/jp034596z

[10] Bentz J L, Olson R M, Gordon M S, Schmidt M W and Kendall R A 2007 *Computer Physics Communications* **176** 589 – 600 ISSN 0010-4655 URL http://www.sciencedirect.com/science/article/B6TJ5-4N7SBTR-1/2/bbe1d382122689b04c012b875569b7b0

[11] Janowski T and Pulay P 2008 *Journal of Chemical Theory and Computation* **4** 1585–1592 (*Preprint* http://pubs.acs.org/doi/pdf/10.1021/ct800142f) URL http://pubs.acs.org/doi/abs/10.1021/ct800142f

[12] Lotrich V, Flocke N, Ponton M, Yau A D, Perera A, Deumens E and Bartlett R J 2008 *The Journal of Chemical Physics* **128** 194104 (pages 15) URL http://link.aip.org/link/?JCP/128/194104/1

[13] Rendell A P, Lee T J, Komornicki A and Wilson S 1992 *Theoretical Chemistry Accounts: Theory, Computation, and Modeling (Theoretica Chimica Acta)* **84** 271–287

[14] Dongarra J, Croz J D, Duff I and Hammarling S 1990 *ACM Transactions on Mathematical Software* **16** 1–17

[15] Bylaska E and et al 2007 *NWChem, A Computational Chemistry Package for Parallel Computers, Version 5.1*

[16] Dunning T H 1989 *The Journal of Chemical Physics* **90** 1007–1023 URL http://link.aip.org/link/?JCP/90/1007/1