A Sustainable Software Innovation Institute for Computational Chemistry and Materials Modeling ((SICM)2)

March 28-30, 2014 Workshop on Portable Parallel Infrastructure

Gary W. Trucks, Gaussian, Inc.

Gaussian, Inc. develops and distributes software for computational chemistry. For us, it is vitally important to consider our user base and target development toward their research needs and to make the best use of the resources they routinely employ. Of course, we also look toward cutting edge and future hardware designs to expand the applicability of electronic structure methods to larger and more complex chemical problems. The challenge in that respect is to identify the types of architectures that will become widespread either due to design or affordability.

Additionally, Gaussian, Inc. is the distributor of LINDA, a tool for parallel programming. We initially partnered with the developers of LINDA for our distributed parallel solution based on proximity and ease of use. Presently, LINDA is supported by Gaussian and though a consulting arrangement with long term developers including Nicolas Carriero. The advantages of LINDA are identical environments across a large range of platforms, the ability to start and stop workers (which has been present for years), interoperability with SMP and the flexibility to use either global shared data or message passing. Because of LINDA's simple approach to parallel code development and also considering our ability to provide effective support to our customers, we do not see a reason to change our current parallel development model.

Our philosophy is to provide the same functionality for every version of Gaussian. This leads to the issue of performance over a wide range of platforms. It is desirable to support one code base across platforms and to minimize specialize code in order to synchronize development and simplify maintainability. Up to this point, we have achieved that while offering both SMP-parallelism within nodes and distributed parallel processing via LINDA. The two can be combined as well to take advantage of several multi-core machines. At present, methods such as Hartree-Fock, CI Singles, DFT energies, optimizations and frequencies are LINDA parallel. MP2 energies and Time Dependent DFT energies and gradients are also LINDA parallel. Portions of MP2 frequencies and Post-SCF methods such as CCSD are LINDA parallel, but other portions are only SMP-parallel. We have attempted to identify widely-used methods and concentrated our efforts to parallelize those algorithms. Our code has the ability to dynamically select algorithms based on available resources (i.e. memory, disk) and decrease the number of threads on the fly if memory resources are not sufficient to efficiently run all available cores. There is also the ability to select algorithms based on FLOP or MOP count.[1]

The limitation of the number of massively parallel machines with hundreds or thousands of cores coupled with restrictions on the user's ability to schedule them for a single job has focused the bulk of our electronic structure code development on more common machines which contain few dozens to

hundreds of processors (either single SMP machines or networks or clusters). This is the type of hardware that is used for the majority of calculations. In turn, our algorithms scale more modestly on larger systems (although they do allow larger systems to be studied). At many computer centers, this is often sufficient because of the number of users performing calculations concurrently. Of course, we continue to improve the parallel performance of our code to track with commodity systems which have become more powerful over the short term.

The pace of hardware development has exceeded software development in our community; however, this also presents the challenge of identifying what types of architectures will persist. It takes substantial time and resources to develop new algorithms and the investment is only worthwhile if the targeted systems have longevity and wide-spread adaption as well as efficient programming tools. For example, algorithms developed for first or second generation GPUs do not exploit the newest hardware. It is necessary to work closely with hardware vendors to synchronize software development with hardware development. One such example is Gaussian, Inc.'s relationship with NVIDIA, Corp. and HP.[2] A crucial component of this collaboration is communication with the compiler group at PGI (now part of NVIDIA, Corp.). The collaborative effort between Gaussian, Inc. and NVIDIA, Corp. has been one of the forces behind the development and implementation of OpenACC in the PGI compilers. OpenACC is a directive-based platform-independent programming model which hides in the compiler most of the complexity in writing code which can run on both the CPU and the GPU. We see these types of collaborative efforts as a necessity to guide development.

At the top of our wish list for parallel development are standard libraries in which the communication is encapsulated, so that MPI, Linda, or whatever can be used for the communication, rather than having MPI wired in throughout the algorithms. Ideally, these would work for very efficiently in SMP environments, but unfortunately most of the presently available standard libraries have abandoned SMP parallelism in favor of treating every core as having a distinct memory system. In order for such libraries to fit our needs, they would have to support a range of operating systems including Windows and MAC OSX. This would both assist with code maintainability and allow for incremental parallel improvements. Moreover, such libraries should never assume unlimited availability of memory and they should give the user full control of any memory allocation performed by the library.

If (SICM)2 is willing to pioneer and share methods for efficient use of massive number of threads, it would be of interest to Gaussian if the work focused on general conclusions applicable to evolving algorithm and did not sacrifice performance on medium-sized collections of processors. The problems we see in this domain include low memory bandwidth, I/O performance, memory available per processor and latency issues. We also welcome all investigation and feedback from (SICM)2 regarding identification of targeted hardware, pace of developments, software tools, communication with hardware developers, etc.

[1]An Improved Criterion of Evaluating the Efficiency of Two-Electron Integral Algorithms, M.J. Frisch, B.G. Johnson, P.M.W. Gill, D.J. Fox and R.H. Nobes, Chem Phys Lett, 206, 225 (1993).

[2] Current Status of Project to Enable Gaussian 09 on GPGPUs , R. Gomperts, M. J. Frisch, G. Scalmani and B. Leback, Spring ACS Meeting, March 2014.