It's going to take more than hardware to advance the state of the art in scientific computation

John M Levesque, Cray Inc

Introduction

The next generation of supercomputers will increase the node performance of the system by several orders of magnitude. Prior to this, Super Computing systems have obtained higher performance by increasing the number of nodes and node performance only increased due to small increases in core on the node. Issues with power consumption are preventing vendors from continuing to design supercomputers with standard commodity chips and the next generation of supercomputers will keep the number of nodes constant and increase performance of the node by either using accelerators or many core nodes that employ energy efficient ways of supplying computational power. This change puts a tremendous burden on application developers. No longer can the supercomputer system be utilized with message passing only. Application developers must start using sophisticated threading on the node and in most cases vectorization of the application is important to achieve the full performance available. The author contends that the typical application developer is not knowledgeable enough to employ sufficient threading and vectorization to effectively utilize the supplied node performance.

Threading on the node

An examination of the applications currently being used on the DOE office of science systems reveals that a large percentage of applications do not currently employee any kind of threading on the node. In most cases the application developers use message passing within the node as well as across nodes. On the new supercomputers that employ more powerful nodes efficient threading is difficult and extremely important. Due to the architectures being employed on the machines the granularity of the threaded region must be large enough to overcome the overhead of initiating parallel threads. While OpenMP is simple to employ on inner loops it is difficult and beyond the capability of the typical application developer to employ at a high level that will achieve good scaling on the node. In addition to obtaining high granularity of the threaded computation it will be increasingly important to efficiently use non-uniform memory on the node. Scaling the parallel threads across a node require the efficient accessing of the memory on the node. As memory organizations become more complicated, the application developer must have a good understanding of the application's use of important arrarys in order to run efficiently

Vectorization on the node

Twenty-five years ago application developers knew how to vectorize their application in order to take advantage of the vector instructions present in all supercomputers of that time. Today the typical application developer completely relies on the compiler to vectorize their application. If the compiler does not vectorize the important looping structures in their application, they assume that their application cannot be vectorized. The techniques to restructure do loops to facilitate the vectorization of complex loops are no longer employed. As the performance gain from vectorization increases it becomes more importance for the application developer to understanding how to write good vectorizable loops

There have been some successes

There have been some successes in refactoring applications to effectively utilize the new accelerated and/or many core systems. Some of these successes have been to completely rewrite the application in a non-portable way. In particular, a large majority of those applications written for utilizing accelerated systems have either been written in Cuda or OpenCL. While some claim that OpenCL is in fact portable currently there are no OpenCL production applications running on DoE Office of Science systems. Another approach that has been successful in several cases is to use a domain specific language (DSL) to isolate the machine dependent optimizations within a library. With this approach, if the library is developed for each targeted platform then the application will be portable.

Other successes have tried using directive based programming approaches. For example, OpenACC and OpenMP 4.0 are directives based programming paradigms that instruct the compiler how to most efficiently generate parallel code for either accelerated system or many core systems. This approach heavily relies on the portability of the directives across the available compilers of parallel systems.

These two approaches 1) using a low level programming approach and the other 2) that relies more heavily on the compiler still requires extensive refactoring of the application in order to produce a program structure that exhibits large granularity parallel regions with low-level vectorizable do loops. This high level refactoring task is very difficult and once again beyond the scope of most application developers.

What is the solution

Unfortunately the problem discussed in this paper cannot be addressed by training alone. One approach that has been highly successful is to establish programming teams consisting of the principal application developer, an expert programmer typically from the vendor's staff and to some extent the user of the application. When optimizing a large application to perform a scientific study on a target super computing system, the characteristics of the problem to be solved dictates how best to refactor the application. With such a team, the expertise in the principal areas that need to be addressed are available and more importantly with the involvement of the application developer, the changes are more likely to be incorporated into future versions of the application. If the target application is an ISV code (Independent Software Vendor), the software vendor is not willing to put in the effort require to optimize the application for supercomputers.

Conclusion

The main thrust of this paper is to convince the community that buying a multi Petaflop system is not enough to advance the state-of-the-art in computational science. Some consideration and funding must be available to prepare important applications for the future. There have been examples where such funding more than pays for itself with the savings in power utilization of the optimize application running in production. (Implementation of COSMO on Accelerators; Oliver Fuhrer, Tobias Gysi, Carlos Osuna, Xavier Lapillonne, Mauro Bianco, Thomas Schulthess, et al)