Adaptive Runtime Systems Coupled with Domain Specific Languages for Computational Chemistry

Laxmikant Kale
Dept. of Computer Science, Univ. of Illinois at Urbana-Champaign
http://charm.cs.illinois.edu kale@illinois.edu

Computational modeling at the molecular level, carried out on supercomputers, promises to bring significant breakthroughs in areas related to chemistry, materials science, and biophysics. However, the upcoming era of ever more powerful machines and increasingly sophisticated algorithms brings with it new challenges, along with these promises. We have been developing an approach based on over-decomposition and adaptive runtime systems that we believe will be extremely useful in this era. Over-decomposition is the idea that the programmer decomposes the computation and data into a number of pieces much larger than the number of processors. This also leads to decomposition into natural logical units of the computation (rather than the processor oriented decomposition). The adaptive runtime system, which is in charge of assigning and reassigning these pieces to processors during execution, continuously instruments and introspects the behavior of the application, and the machine. It then deploys several run-time strategies including dynamic load balancing, to improve the performance of the running application. In addition to improving performance, this also leads to efficient composition of independently developed parallel modules. These ideas have been embodied in the parallel programming system called Charm++.

The Exascale era will be characterized by variability and heterogeneity at the machine level. Different types of processors will co-exist, possibly within a single chip. E.g. accelerators with higher energy efficiency that are useful for certain types of algorithms will co-exist with many-core and multi-core processors, along with possibly custom hardware. Another type of heterogeneity: Thermal considerations and process variations may mean processors running at different speeds that are dynamically changing. Component failures are likely to become common occurrence. The over-decomposition based methodology identified above will stand us in good stead in this unsteady landscape.

We have demonstrated in Charm++ the ability to balance load dynamically, tolerating faults (the computation continues to run as the runtime system detects faults and takes corrective actions based on in-memory checkpoints, and/or message logs), and responding to changes in core temperatures and minimizing/optimizing energy.

At the same time, the community is looking at developing more sophisticated algorithms. There is a need to compose independently developed parallel modules effectively. The message-driven execution engendered by the overdecomposition-based programming model allows for effective interoperability between modules; e.g. modules can overlap execution with each other, thus using idle time in one module to carry out computations of the other module automatically.

We have demonstrated the Charm++ methodology in multiple scalable applications in wide use among scientists. Within the focus domains of this community, we collaboratively developed two applications of interest: NAMD for biophysics and OpenAtom for materials modeling (a Car-Parinello MD). In addition, Charm++ is being used for applications in astronomy, simulation of epidemics, etc. Newer frameworks such as *MADNESS,* as well as several aspiring new exascale languages, also argue for an adaptive runtime system.

The Charm++ approach has been to develop the abstractions in a bottom up manner, so that the system is always pragmatic and usable by practitioners, even as we continually add capabilities and increase the level of abstraction continually. I'd like to appeal to the computational chemistry to consider using such novel systems (specifically those based on over-decomposition) to start becoming future-proof.

A second frontier the computational chemistry community needs to open involves improving programmer productivity. Today, it takes a huge, almost formidable, effort to develop a new parallel code. The basic algorithmic ideas (which take parallel efficiency into account) need to be expressed all the way at the lowest level of MPI or Charm++ (although Charm++ automates resource management, it still is a relatively low level language, as far as expressing interactions is concerned). However, there are a relatively small number of data structures and operations that can cover a large number of algorithms in use in chemistry. This argues for development of domain specific languages. The tensor contraction engine (Sadayappan et al) is an example of such an abstraction. I imagine a day when the computational chemist is able to express algorithms almost at the mathematical level, while a compiler translates them into an effective parallel program. However, such a system must allow for expression of intermediate levels of abstraction for the sake of efficiency, if it is to be useful.

As a simple example, the OpenAtom code has a step that does "orthonormaliztion" of electronic states represented in reciprocal space. A part of this involves constructing a matrix (S) which has an entry for each pairs of states. It is easy to express the S matrix mathematically, but for efficient computation, an intermediate data structure that holds pieces of the product of corresponding terms of the two states is necessary. The compiler will find it very difficult to infer the existence of this intermediate object by itself. What we should aim at is a language (for this sub-domain) that allows us to express the idea of the intermediate data structure, but automate the mechanics of populating and using this data structure in every iteration. We seek the help of the community to present case studies and identify potential abstraction levels and features.

To summarize: a parallel language based on adaptive runtime system, and a collection of domain-specific languages and frameworks are two approaches the community should combine to address the challenges of exascale machines, and to reap the benefits arising from them.