Composable and modular Exascale Programming Models with intelligent runtime systems

Laxmikant (Sanjay) Kale
Parallel Programming Laboratory (http://charm.cs.uiuc.edu)
University of Illinois at Urbana-Champaign

Exascale machines, and the kinds of applications that are expected to run on them, pose significant challenges for the programming models community. Significant increase in the degree of parallelism within a node, heterogeneity, the sheer number of computing elements available, as well as considerations of power and resilience create challenges and opportunities for the design of a new generation of programming models. I will outline a few broad imperatives, some of which require radical changes for such designs.

To begin with, I will argue that it would be beneficial to (almost) eliminate the notion of "processor" from the ontology of the programmer. The program should be expressed in terms of the work-units and data-units of the application without reference to which processor they are housed on. This can be supported by an automated resource management system which can adapt to variations in the application configurations/evolution, as well as to variations in the machine environment. At exascale, the programming models will have to depend on novel, highly powerful and intelligent adaptive runtime systems. These systems will monitor multiple aspects of an ongoing execution, and take corrective actions in order to optimize multiple criteria such as performance, power, and effect recovery from component failures. Further, parallel composition of multiple, independently developed modules must be strongly supported, without explicitly requiring partitioning processors or sequencing modules. Message-driven execution is a powerful runtime mechanism that, in my opinion, is essential for supporting such modularity.

Such modularity is impossible without support for interoperability between programming models, including legacy models such as MPI and OpenMP. The runtime system can tie these modules together and support communication between disparate models, allowing the programmer to use whichever model is best suited to each portion of an application. Interoperability between models will also provide an opportunity to simplify parallel programming by introducing simple, incomplete programming models which may be incapable of expressing arbitrary parallel interactions or be limited to specific data structures but which provide increased safety and simplicity without sacrificing performance. I expect the exascale toolbox to eventually include a small set of interoperable incomplete models designed to address common problems encountered in parallel application development. We need to develop new data-structure specific frameworks, constructed from the point of view of interoperability. Compiler support will be required, not for automatic parallelization, but for supporting convenient syntax and basic analysis aimed at optimization.

Finally, I argue strongly against premature standardization of programming models. If we support interoperability via adaptive runtimes, we will enable the broad experimentation that is necessary to create a powerful and flexible toolbox of multiple programming models.

--

Laxmikant (Sanjay) Kale
Professor, Computer Science
201 N. Goodwin Avenue
Urbana, IL 61801-2302

http://charm.cs.uiuc.edu
kale@illinois.edu
Ph: (217) 244-0094
FAX: (217) 265-6582