Evolving MPI+X Toward Exascale

This installment of Computer's *series highlighting the work published in IEEE Computer Society journals comes from* IEEE Transactions on Parallel and Distributed Systems.

David A. Bader, Georgia Tech

The recent trend in high-performance computing (HPC) to adopt accelerators such as GPUs, field-programmable gate arrays, and coprocessors has led to significant heterogeneity in computation and memory subsystems. Application developers typically employ a hierarchical message passing interface (MPI) programming model across the cluster's compute nodes, and an intranode model such as OpenMP or an accelerator-specific library such as compute unified device architecture (CUDA) or open computing language (OpenCL) for the CPUs and accelerator devices within each compute node. To achieve acceptable performance levels, application programmers must have in-depth knowledge of machine topology, compute capability, memory hierarchy, compute-memory synchronization semantics, and other system characteristics. However, explicit management of computation and memory resources along with a disjointed programming model mean that programmers must make tradeoffs between performance and productivity.

In "MPI-ACC: Accelerator-Aware MPI for Scientific Applications" (*IEEE Trans. Parallel and Distributed Systems*, vol. 27, no. 5, 2016, pp. 1401–1414), Ashwin Aji and his colleagues from Virginia Tech, Argonne National Laboratory, North Carolina State University, and Rice University present a unified programming model and runtime system for HPC clusters with heterogeneous computing devices. Specifically, they introduce MPI-ACC, an evolutionary step in the MPI+X programming model, which is the de facto standard for distributed memory clusters. By evolving an already popular programming model, the authors make it easier to modernize the code of existing MPI-based applications.

Aji and his team note that when invoking a data-movement routine in MPI-ACC, programmers can simply describe additional data attributes specific to the within-node elements—such as the GPU command queue, execution stream, or device context—without changing the MPI standard. MPI-ACC's runtime system employs user-specified data attributes to not only perform end-to-end data movement across the network but also synchronize with in-flight GPU kernels to achieve efficient overlap of communication with computation. The authors contrast their simple descriptive approach with the complex prescriptive approach of existing GPU-aware MPI implementations. They argue that although other approaches provide end-to-end data movement support between GPUs, they don't have a mechanism to express the data's execution attributes, which puts the burden of overlapping communication with computation on end users.

The investigators also performed in-depth analysis of how MPI-ACC can be used to scale in-production scientific applications such as an epidemic spread simulation and a seismology simulation. They further show that the MPI-ACC's pipelined end-to-end data movement, scalable intermediate resource-management techniques, and enhanced execution progress engine outperform baseline implementations that use MPI and CUDA separately.

MPI-ACC evolves and unifies the MPI+X programming model. Its expressive and familiar interface allows programmers to describe the appropriate computation or communication targets, and its runtime system automatically achieves efficient cluster utilization.

DAVID A. BADER is a professor and chair of the School of Computational Science and Engineering, College of Computing, Georgia Tech. Contact him via www.cc.gatech.edu/~bader.

Selected CS articles and columns are also available for free at **http://ComputingNow.computer.org**.

0018-9162/16/$33.00 © 2016 IEEE