Infrastructure and API Extensions for Elastic and Fault Tolerant Execution of MPI Applications

Isaías A. Comprés
compresu@in.tum.de

Michael Gerndt
gerndt@in.tum.de

Abstract

High Performance Computing (HPC) systems are traditionally shared among several research institutions. A resource manager with a scheduler that operates in space sharing mode is typically used. Space sharing gives applications exclusive access to the resources for their entire execution. This approach enables a very stable and predictable environment to applications.

Future exascale systems will be much more dynamic due to increased failure rates [1] and power consumption variability in the hardware [2]. Furthermore, applications are becoming more dynamic. For example, in a Tsunami simulation the number of grid points can dramatically increase while the wave is propagating from the epicenter to the coasts; in such a case, the computational power and the amount of memory provided to the application need to increase dynamically. Static resource management will be inadequate [3] for such advanced systems and applications.

This paper proposes an extension to the MPI standard. The extension consists of four new operations that allow for the dynamic modification of the number of processes of an application's world communicator, matching any changes in its resources at runtime. These operations offer better support for moldable and malleable applications on large scale distributed memory systems.

A short description of each proposed operation is presented here:

- `MPI_INIT_ADAPT`: Initializes the library in adaptive mode and indicates the status of the local process: new or joining. A process is new when it was created as part of the initial application launch, or joining when it is part of an expansion.
- `MPI_PROBE_ADAPT`: Indicates whether the application is required to adapt. If an adaptation is required, it also provides the status of the preexisting process. The status can be staying or retreating.
- `MPI_COMM_ADAPT_BEGIN`: Begins the adaptation window. This operation provides a set of helper communicators that allow the application to reach any preexisting or joining processes.
- `MPI_COMM_ADAPT_COMMIT`: Completes the adaptation window. All preexisting processes that did not retreat and all joining processes are members of the `MPI_COMM_WORLD` communicator after this operation.

A prototype solution (based on the MPICH library and the SLURM resource manager) is presented and evaluated with a pair of elastic scientific applications that make use of the new MPI extensions. The cost of the new MPI operations is shown to be negligible due mainly to the latency hiding design, leaving the application's adaptation time as the only significant performance cost.

References

[1] Thakur, R., Balaji, P., Buntinas, D., Goodell, D., Gropp, W., Hoefler, T., Kumar, S., Lusk, E. and Trff, J.L., 2010. MPI at Exascale. Proceedings of SciDAC, 2, pp.14-35.

[2] Rountree, B., Ahn, D.H., De Supinski, B.R., Lowenthal, D.K. and Schulz, M., 2012, May. Beyond DVFS: A first look at performance under a hardware-enforced power bound. IPDPSW, 2012 (pp. 947-953). IEEE. Vancouver

[3] Thakur, R., Balaji, P., Buntinas, D., Goodell, D., Gropp, W., Hoefler, T., Kumar, S., Lusk, E. and Trff, J.L., 2010. MPI at Exascale. Proceedings of SciDAC, 2, pp.14-35. Vancouver