DISP: Optimized Initialization for Scalable MPI Startup

Significance and Impact: We have developed a new scalable startup scheme with three internal techniques, namely Delayed Initialization, Module Sharing and Prediction-based Topology Setup (DISP). The DISP scheme greatly benefits the collective initialization of the Cheetah module and helps boost the performance of non-collective initialization in the Tuned module. Evaluation of our implementation on Titan with up to 4096 processes show that our delayed initialization can speed up the startup of Tuned and Cheetah by an average of 32.0% and 29.2%, respectively. The module sharing can reduce the memory consumption of Tuned and Cheetah by up to 24.1% and 83.5%, respectively, and our prediction-based topology setup can speed up the startup of Cheetah by up to 80%.

Research Details:

- We have examined and analyzed the performance and scalability issues of the MPI startup.
- We have proposed a hybrid solution (Delayed Initialization, Module Sharing and Prediction-based Topology Setup) that includes a suite of optimizations to address the startup problem.
- We have implemented our solution and conducted experiments on Titan and demonstrate the performance benefits.

Sponsor/Facility: Work was performed at ORNL and Florida State University.

PI and affiliation: David Bernholdt– Oak Ridge National Laboratory

Team: Manjunath Gorentla (ORNL), Swaroop Pophale (ORNL), Huansong Fu (FSU) and Weikuan Yu (FSU)

Publication: Huansong Fu, Swaroop Pophale, Manjunath Gorentla Venkata, and Weikuan Yu. 2016. DISP: Optimizations towards scalable MPI startup. In *Proceedings of the First Workshop on Optimization of Communication in HPC* (COM-HPC '16). IEEE Press, Piscataway, NJ, USA, 53-62. DOI: https://doi.org/10.1109/COM-HPC.2016.11

Overview:

DISP is a suite of optimizations towards scalable MPI startup. In contrast to the conventional full-fledged, non-reusable, and communication-intensive initialization that has been revealed to be extremely costly in terms of both execution time and memory consumption, DISP initializes the communicators only partially, offloads the laborious part of initialization to where they are needed, reuse collective modules whenever possible, and only resort to collective communication for initialization when necessary. Our solution features a combination of three techniques. Firstly, we design a *delayed initialization* scheme that postpones the completion of communicator initialization to the point where the communicator is actually used. This replaces existing communicator creation routines where the communicators are fully initialized at the beginning; the delayed initialization only initializes a *shadow communicator* on top of a full- fledged communicator. The shadow communicator abstracts the essential subset of the communicator's information. The communicators are then initialized on an *on-demand* basis, only to be fully initialized when it is used for the first time by a collective operation. Secondly, we have both *temporal* and *spatial* module sharing mechanisms that realize the sharing of communicator modules between either different communicator objects of one process, or several processes that reside on the same node. The eligibility criteria of two communicators for module sharing are that their shadow communicators must be identical. Lastly, we improve the MPI startup with a mechanism called *prediction-based topology setup* that can locally compute most of the predictable hierarchical topology needed by Cheetah initialization without communicating back and forth over the network. Our design requires no changes to MPI APIs or the user application.