Advanced MPI Programming

Julien Langou[1] and George Bosilca[2]

[1] Department of Mathematical Sciences,
University of Colorado, Denver, CO, USA
langou@math.cudenver.edu
[2] Innovative Computing Laboratory,
University of Tennessee, Knoxville, TN, USA
bosilca@cs.utk.edu

MPI provides a large range of features allowing various approaches for parallel computing. This tutorial will present interesting features from the MPI-1 standard. These features extend the user knowledge about MPI way beyond the few basic standard functions, giving them the opportunity to implement better, simpler and potentially faster parallel algorithms. This tutorial will cover several features from medium level to advanced of the MPI-1 standard to enable users to exploit fully MPI.

Identifying the bottlenecks. Once an initial version of a parallel application exists, the first question raised is how to improve it. While a large improvement might come from algorithmic modifications, there is usually a large space of improvements that might come from a correct useage of the MPI specification. Before starting any optimization, the user has first to identify the portions of the application that have the greatest potential for improvement. The basic approach is to use the MPI profiling layer. Unfortunately, this layer lacks the transparency required to show the effective data transfer over the network, missing a rich set of information about the application behavior. PERUSE is a more suitable layer to get access to all the hidden information available from the MPI libraries.

Optimizing memory accesses. The MPI specification introduces a number of functions for creating MPI data-types. These data-types represent memory layouts and can be used for message exchange; both in point-to-point or collective communications. Choosing the right data-type is a performance critical choice as its impact on the communication performance is tremendous.

Advanced point-to-point communications. While blocking communications are easier to masterize, they introduce one synchronization stage per communication which might be harmful to the overall performance of the application. Moving away from this blocking model, by using non-blocking point-to-point communications, usually allows the user for a better overlap between multiple communications leading to a shorter execution time of the parallel application.

Collective communications. The collective communications provides a highly optimized variety of algorithms for global data exchange between a set of MPI processes. These algorithms cover some of the most generic collective communications such as broadcast, barrier, etc. Some of them (i.e. those derived from the reduce operation) can be extended using user-provided operations.

For each item in the above list, we will provide attendees with small and comprehensible example codes that have been identified as critical part of larger relevant application codes. Starting from these small codes, we will show how to exploit MPI functionallities in order to improve simplicity and efficiency of the application. The example codes and their modifications will be made available to the attendees.